

SUPER

Anno 2
n. 7 - Lug./Ago. 1985

L. 7.500

Spedizione in
abbonamento
postale Gruppo III/70

7

SUPERCOMMODORE
la rivista per gli utenti
dei prodotti
COMMODORE

C-COMMODORE

Una pubblicazione della

J.soft EDITRICE

In collaborazione con

GRUPPO
EDITORIALE
JACKSON



**NO ZAP:
UN PROGRAMMA
ANTI BLACK-OUT**

**3D COLOR MAZE
PER C16**

**METABASIC
PER C64**

**ADSR
PER VIC 20**

**POOL
PER VIC 20 E C64**

Ritorna in edicola VIDEO BASIC

Il corso più entusiasmante su cassetta
del Gruppo Editoriale Jackson per Commodore 64,
VIC 20 e Spectrum

200.000 copie vendute

del 1° fascicolo della prima edizione

Ogni lezione
uno spettacolo

Col 1° fascicolo
una cassetta giochi



Il corso è composto da:
20 fascicoli + (Quattordicinali)
20 cassette +
5 splendidi raccoglitori

Oggi è davvero facile imparare il Basic. Con Video Basic il corso su cassetta che ti permette di programmare subito il tuo computer. È facile: tu chiedi, lui risponde, tu impari. Passo dopo passo. Sul tuo schermo appaiono le domande, le risposte, gli esercizi e

tu, senza fatica, presto e bene, impari a conoscere e programmare il tuo computer, sia esso un VIC 20, un Commodore 64 o un Sinclair. Video Basic è in edicola. Provalo subito. Ogni lezione è uno spettacolo.

Oggi il Basic si impara così. Video Basic, il corso su cassetta per parlare subito col tuo computer.

Video Basic
per imparare non solo il Basic.



Un'altra grande idea firmata
GRUPPO EDITORIALE JACKSON
Milano-San Francisco-Londra-Madrid

SUPER

COMMODORE

5 EDITORIALE POINT

**DOVE VA L'HOME
COMPUTER?**

di **Pietro Dell'Orco**

7 READ & WRITE

LA POSTA DEI LETTORI

10 OPEN

**A CHE SERVE L'HOME
COMPUTER**

di **M. Cristuib Grizzi**

13 LA CULTURA ...then

**ABC DEL LINGUAGGIO
MACCHINA**

Parte quarta
di **U.G. Barzaghi**

30 REM:HW

**SCOPRIAMO INSIEME
LA STAMPANTE
MP8-802**

di **E. Sarcina**

36 LOAD

**36 NO ZAP: UN
PROGRAMMA
ANTI BLACK-OUT**
di **J. Blake Lambert**
trad. e adatt.
di **M. Cristuib Grizzi**

**40 3D COLOR MAZE
PER C16 E PLUS/4**
di **A. Motta**

44 APPLE HUNT
di **J. Arlet**
trad. e adatt.
di **M. Cristuib Grizzi**

51 METABASIC PER C64
di **K. Mykytyn**
trad. e adatt.
di **M. Cristuib Grizzi**

64 POOL
di **J.T. Woyton**
trad. e adatt.
di **M. Cristuib Grizzi**

76 NUMBER QUEST
di **G. e B. Schueler**
trad. e adatt.
di **M. Cristuib Grizzi**

80 HEAT SEEKER
di **J. Wolverson**
trad. e adatt.
di **M. Cristuib Grizzi**

91 VICsound

**ADSR: CON IL VIC IL
SUONO DEL C64**
di **V. Silvestroni**

94 PRINT

RECENSIONI LIBRI
di **M. Cristuib Grizzi**

96 Input-output

PICCOLI ANNUNCI

J. soft s.r.l.
**DIREZIONE, REDAZIONE,
AMMINISTRAZIONE**
Viale Restelli, 5
20124 MILANO
Tel. (02) 68.88.228

DIRETTORE RESPONSABILE:
Pietro Dell'Orco

COORDINAMENTO TECNICO:
Riccardo Padellaro

REDAZIONE:
Mauro Cristuib Grizzi

**HANNO COLLABORATO A
QUESTO NUMERO**
U.G. Barzaghi
A. Motta
E. Sarcina

GRAFICA E IMPAGINAZIONE:
Margherita La Noce
Ivana Rossi
Raffaella Toffolatti

FOTOCOPIAZIONE:
Graphotek - Via Astesani, 16
Tel. 02/64.80.397
20161 MILANO

CONTABILITÀ:
Giulia Pedrazzini
Flavia Bonatti

**AUTORIZZAZIONE ALLA
PUBBLICAZIONE:**
Tribunale di Milano n° 201
del 14.04.1984

STAMPA:
Litografia del Sole
Albairate (MI)

PUBBLICITÀ
Concessionaria per l'Italia e
l'Estero J. Advertising s.r.l.
Viale Restelli, 5
20124 MILANO
Tel. (02) 68.82.895-68.80.606-68.87.233
Tlx. 316213 REINA I

Concessionaria esclusiva per la
DIFFUSIONE in Italia e Estero:
SODIP - Via Zuretti, 25
20125 MILANO

Spedizione in abbonamento
postale Gruppo III/70
Prezzo della rivista L. 7.500
Numero arretrato L. 15.000
Abbonamento annuo: L. 82.500
(11 numeri con cassetta);
L. 38.500 (11 numeri senza cassetta);
per l'estero: L. 110.000 (11 numeri con
cassetta) - L. 50.000 (11 numeri senza
cassetta)

I versamenti devono essere indirizzati a:
J. soft s.r.l.
Viale Restelli, 5 - 20124 MILANO
mediante emissione di
assegno bancario, cartolina
vaglia o utilizzando
il c/c postale n. 19445204.
Per i cambi di indirizzo indicare,
oltre al nuovo, anche l'indirizzo
precedente ed allegare alla
comunicazione l'importo di L. 500,
anche in francoboli

© TUTTI I DIRITTI DI
RIPRODUZIONE O TRADUZIONE
DEGLI ARTICOLI PUBBLICATI
SONO RISERVATI

GLI ARTICOLI TRADOTTI SONO
TRATTI DALLA RIVISTA COMPUTE!
O DA COMPUTE! GAZZETTE -
COMPUTE! PUBLICATION, INC.
P.O. BOX 5406,
GREENSBORO, NC 27403 - USA

La piccola guida del principiante

Che cos'è un programma?

Il solo computer non può compiere alcuna operazione. Un computer possiede potenzialità, ma, come una macchina senza benzina, senza un programma non può funzionare. La maggior parte dei programmi pubblicati su **SUPERCOMMODORE** per i computer Commodore sono scritti in un linguaggio per calcolatori chiamato BASIC. Il BASIC è facile da imparare ed è disponibile, di serie, nel VIC 20 e nel Commodore 64.

Programmi BASIC

Ogni mese **SUPERCOMMODORE** pubblica programmi sia per il VIC 20 che per il C64. Tanto per cominciare, se possiedi un VIC 20 copia solo i programmi scritti per la tua macchina, indicati con "versione per VIC 20". Più tardi, quando avrai acquisito esperienza con il BASIC del tuo computer, potrai cercare di copiare e convertire programmi scritti per altri calcolatori. Diversamente dal linguaggio corrente, che può essere variamente interpretato, il BASIC di solito ha un solo modo corretto per indicare qualcosa. Ogni lettera, carattere o numero ha il suo significato. Un errore banale è costituito dalla sostituzione del numero "0" con la lettera "O" oppure il carattere minuscolo "l" invece del numero "1" o ancora il carattere maiuscolo "B" con il numero "8". Devi anche inserire tutta la punteggiatura, i due punti (;) e le virgole, copiando esattamente ciò che appare sulla rivista. Gli spazi possono essere importanti. Per essere sicuro copia il listato esattamente come si presenta.

Le parentesi e i caratteri speciali

L'eccezione per questa regola di copiatura si presenta quando incontrerai indicazioni tra parentesi, quali: "(GIÙ)". Ogni cosa compresa tra parentesi è un carattere speciale oppure un carattere che non può essere facilmente prodotto con la stampante. Incontrando un carattere di questo tipo fai riferimento alla "Piccola guida per l'input dei programmi".

Le istruzioni DATA

Alcuni programmi contengono una sezione, o delle sezioni, di istruzioni DATA. Queste linee di istruzione forniscono le informazioni di cui il programma ha bisogno. In alcuni casi le istruzioni DATA costituiscono il programma vero e proprio, in altri contengono codici grafici. Queste

linee sono particolarmente soggette agli errori. Se un solo numero in una linea di istruzione DATA è sbagliato, il calcolatore potrebbe "pantarsi" o distruggere il programma. La tastiera e il tasto STOP appaiono inattivi e lo schermo completamente vuoto. Non lasciarti prendere dal panico. Non si è verificato alcun danno.

Per riprendere il controllo devi spegnere il computer e successivamente riaccenderlo. Ciò cancellerà qualsiasi programma presente in memoria, per cui è sempre necessario fare il SAVE del tuo programma prima di comandare il RUN.

Se il computer si ferma, puoi caricare (LOAD) il programma e cercare l'errore.

A volte, quando il programma viene "lanciato", un'istruzione DATA errata può causare un messaggio di errore.

Il messaggio di errore potrebbe riferirsi alla linea di programma che legge (READ) il contenuto delle istruzioni DATA.

Come conoscere il computer

Dovresti prendere confidenza con il computer prima di procedere alla copiatura del programma.

Impara le istruzioni che si usano per memorizzare e richiamare i programmi da nastro o da disco. Dovrai conservare una copia del tuo programma, se non vorrai copiarlo ogni volta che lo devi usare. Impara ad usare le funzioni di "editing" della tua macchina. Come puoi correggere un errore? Puoi sempre ricopiare la linea e in questo caso devi sapere come procedere. Sapresti come inserire i caratteri in "inverse", i caratteri minuscoli e quelli di controllo?

Tutto ciò è spiegato nel manuale del calcolatore.

Un veloce ripasso

- 1) Copia il programma una linea alla volta, con ordine. Premi RETURN alla fine di ogni linea. Usa il tasto "DEL" per correggere gli errori.
- 2) Confronta la linea copiata con quella presente nella rivista. Puoi controllare l'intero programma nel caso in cui si presenti un errore quando esegui il RUN.
- 3) Accertati di aver inserito le istruzioni tra parentesi graffe con gli appropriati caratteri di controllo (fai riferimento alla "Piccola guida per l'input dei programmi" che trovi in questa stessa rivista).

Siamo spiacenti di non poter rispondere singolarmente alle richieste di informazioni circa i programmi, prodotti o servizi apparsi su **SUPERCOMMODORE**.

POINT



Dove va
l'home computer?

Giungono dagli Stati Uniti voci allarmistiche circa il presente e il futuro degli home computer e dei piccoli personal computer.

Ci sono fabbriche che hanno dovuto cessare temporaneamente la produzione delle macchine per smaltire le scorte di magazzino.

Forti le critiche mosse a proposito del modo di pubblicizzare la presunta indispensabilità del computer nella casa oltre alle motivazioni, a volte senza troppo senso, con cui tutte le case produttrici hanno cercato di imporre a chiunque l'attrezzo tecnologico.

Da sempre l'Italia si è mossa, ovviamente con tempi diversi, sul modello americano; si sono infatti riscontrate le stesse tendenze del mercato statunitense circa la diffusione, l'utilizzo e quant'altro attiene al mondo del personal computing.

Ora dunque, in linea teorica, anche da noi c'è da attendersi una recessione che potrebbe contrarre la domanda e l'offerta di questi prodotti.

Ad una attenta analisi però si possono riscontrare sostanziali differenze di diffusione, facendo comunque le debite proporzioni, tra il mercato italiano e quello americano.

Sono svariati gli esempi che possono essere citati a conferma della ancora scarsa diffusione del personal nel nostro paese.

Un esempio per tutti potrebbe essere rappresentato dalla situazione nelle scuole dove, nonostante le pressioni esercitate da più parti il calcolatore non è ancora stato adottato massicciamente quale indispensabile ausilio didattico.

Pare che da noi, oltre agli impieghi in ambiti ben precisi, il calcolatore non abbia ancora trovato una precisa collocazione, del resto una delle principali caratteristiche risiede proprio nella possibilità di trasformarsi per assumere ruoli spesso molto diversi tra loro. Ogni utilizzatore deve cercare il migliore modo di sfruttare il proprio "attrezzo" ponendosi sempre precisi obiettivi da raggiungere.

Non ci stancheremo mai di ripetere che non bisogna subire passivamente il calcolatore dandogli da elaborare ogni cosa che capita di trovare pubblicata, ma usare quel materiale come spunto creativo per elaborare nuove realizzazioni.

Siamo convinti che siano ancora moltissimi gli utenti di "prima mano" o persone che aspirano al possesso del personal computer, quindi ancora tanto lavoro da fare anche da parte nostra.

Dunque, per la realtà del nostro paese, la recessione americana non ci deve preoccupare più di tanto; da noi c'è ancora tanto da fare e, forse, gli home e i personal computer non hanno ancora conosciuto la loro alba migliore.

Per finire tutta la redazione augura ai lettori buone vacanze, raccomandando a tutti di non lasciare a casa, o in disparte troppo a lungo, il fedele "amico".

Pietro Dell'Orco

Per collaborare a SUPERCOMMODORE

La rivista è interessata ad articoli e programmi riguardanti la linea Commodore, di interesse generale, chiari ed esaurienti. Vi proponiamo una piccola "guida", che ha particolarmente lo scopo di rendere più accurata la stesura dei vostri manoscritti: seguendo i nostri consigli si accresceranno le probabilità che le vostre buone idee e i vostri programmi vengano pubblicati.

1 l'angolo superiore sinistro della prima pagina dovrà contenere: nome, cognome, indirizzo, numero telefonico, codice fiscale, data di spedizione, luogo e data di nascita.

2 l'angolo superiore destro della prima pagina dovrà contenere la marca e il tipo di computer al quale il lavoro si riferisce, unitamente ad eventuali espansioni di memoria o periferiche richieste.

3 il titolo dell'articolo, sottolineato, dovrà iniziare a circa due terzi in altezza della prima pagina.

4 le pagine seguenti potranno essere battute normalmente, con la condizione che l'angolo superiore destro contenga un'abbreviazione del titolo e del cognome, unitamente al numero di pagina. Per esempio, Sprite Ed.../Brambilla/2.

5 il testo dell'articolo dovrà essere battuto a macchina con interlinea di uno spazio e mezzo, massimo due spazi; un margine di almeno un centimetro dovrà essere lasciato su entrambi i lati dello scritto.

6 dovranno essere usati fogli in formato UNI A4 (cm. 21 x 29,7) e il testo, scritto in caratteri maiuscoli e minuscoli, dovrà occupare una sola facciata del foglio.

7 nel caso il testo comprenda più fogli, questi dovranno essere uniti con un fermaglio o con un punto metallico aperto.

8 avendo intenzione di spedire più di un articolo, questi dovranno essere inviati separatamente insieme alla rispettiva copia su supporto magnetico.

9 programmi brevi (meno di 20 linee) potranno essere inseriti nel testo, mentre programmi più lunghi dovranno essere listati separatamente. È **ESSENZIALE** per noi disporre di una copia del programma registrata più volte su supporto magnetico, su entrambi i lati dello stesso. È preferibile usare nastri di buona qualità e di lunghezza non eccessiva; la cassetta o il disco dovranno essere etichettati con il nome dell'autore, il titolo dell'articolo, il computer interessato e soprattutto le eventuali espansioni richieste. Come sug-

gerimenti di programmazione si consiglia di usare le istruzioni "CHR\$(x)", "TAB(x)", "SPC(x)", piuttosto che stringhe di manipolazione del cursore.

Ad esempio, per uno scroll di 5 linee l'istruzione "FORI = 1 TO 5:PRINT:NEXT". è molto più interpretabile di 5 Q inverse; e, invece di una dozzina di simboli di cursore a destra, perché non usare semplicemente "PRINT SPC(12)"? Un rapido controllo dei programmi per operare queste sostituzioni sarà molto apprezzato da noi e dai lettori.

10 per maggior chiarezza, all'interno dell'articolo è conveniente usare caratteri maiuscoli riferendosi a istruzioni BASIC (esempio RETURN, LIST, RND, PRINT ecc.). Se si desidera evidenziare una parola, è preferibile sottolinearla piuttosto che scriverla in carattere maiuscolo.

11 gli articoli ed i programmi potranno avere qualsiasi lunghezza: da una routine di una sola linea fino a programmi molto complessi.

12 volendo includere diapositive, queste dovranno avere formato 24x36, o 6x6.

13 non prenderemo in considerazione articoli che siano stati sottoposti ad altre case editrici.

14 il materiale non pubblicato non verrà restituito.

15 il compenso per la collaborazione prestata sarà commisurato alla complessità e all'interesse del testo e/o del programma (da un minimo di L. 50.000 a un massimo di L. 300.000). Il pagamento verrà effettuato in caso di pubblicazione del lavoro.

Inviare idee e programmi a:
SUPERCOMMODORE
Viale Restelli, 5
20124 Milano

e saremo lietissimi di pubblicare i contributi migliori.

La Redazione

alle aziende produttrici di software, e quindi evitare il diffondersi di copie "pirata" dei programmi, questi ultimi vengono in genere protetti contro la duplicazione. Il tipo di protezione utilizzato varia da azienda ad azienda, ed addirittura da programma a programma nell'ambito della stessa azienda, dal momento che la tecnica di protezione viene in genere sviluppata dagli autori del programma stesso e non dall'azienda che lo commercializza.

La protezione può consistere in un semplice autostart ottenuto tramite linguaggio macchina ed inibizione dei tasti RUN/STOP e RESTORE, oppure può essere più sofisticata e controllare il contenuto di certe locazioni di memoria: se questo non è il valore corretto, significa che è stata compiuta qualche manipolazione, ed il computer si resettava perdendo il contenuto della memoria.

Naturalmente, non tutti sono "pirati", ed è più che lecito desiderare di crearsi una copia di back-up, o di velocizzare il caricamento di un lungo programma che si è regolarmente acquistato.

Le tecniche da utilizzarsi a questo fine richiedono però una buona conoscenza del computer ed almeno una certa dimestichezza con il linguaggio macchina e la mappa di memoria: bisogna innanzi tutto capire come funziona la protezione (che, come detto, è diversa caso per caso), quindi aggirarla con l'uso di un buon assembler/disassembler come il "Supermonitor" da noi pubblicato recentemente.

La cosa importante è capire se vi sia o meno un caricatore, che precede il programma vero e proprio, ed analizzarlo attentamente. E quasi sempre quest'ultimo che provoca l'autostart ed in genere - se in BASIC - contiene un'istruzione SYS. Bisogna annotarsi il numero di questa SYS e quindi leggere gli indirizzi iniziale e finale delle locazioni di memoria occupate dal programma vero e proprio. Per ottenere questo scopo occorre posizionarsi con il nastro all'inizio del programma e digitare il seguente programma:

```
100 PRINT"[CLR]CARICAMENTO L/M...":GOSUB 30
110 PRINT"PREPARARE IL NASTRO"
120 INPUT"NUMERO DEL FILE":F$
130 TS=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(TS)
140 POKE782,ZK/256
150 POKE781,ZK-PEEK(782)+256*PEEK(780),LEN(TS)
160 POKE780,1:POKE781,1:POKE782,1:SYS6546
170 SYS49152
180 W2=PEEK(174)+256*PEEK(175)
190 W1=PEEK(193)+256*PEEK(194)
190 SM=PEEK(641)+256*PEEK(642)
200 PRINT"[GIU']QUESTO PROGRAMMA INIZIA A :W1
210 PRINT"E FINISCE ALLA LOCAZIONE :W2
220 END
230 LM=49152
240 FOR I=LNTOLM-256:READA:CK=A:POKEI,A
250 IFCK<>35693THENPRINT"ERRORE NELLE DAT
260 PRINT"O.K.":RETURN
270 DATA 133,147,169,0,133,144,165,186
280 DATA 208,3,76,19,247,201,3,240
290 DATA 249,144,123,164,183,208,3,76
300 DATA 16,247,166,185,32,175,245,169
310 DATA 96,133,185,32,213,243,165,186
320 DATA 32,9,237,165,185,32,199,237
```

```
330 DATA 32,19,238,133,174,165,144,74
340 DATA 74,176,80,32,19,238,133,175
350 DATA 138,208,8,165,195,133,174,165
360 DATA 196,133,175,32,210,245,169,253
370 DATA 37,144,133,144,32,225,255,208
380 DATA 3,76,51,246,32,19,238,170
390 DATA 165,144,74,74,176,232,138,164
400 DATA 147,240,12,160,0,209,174,240
410 DATA 8,169,16,32,28,254,44,145
420 DATA 174,230,174,208,2,230,175,36
430 DATA 144,80,203,32,239,237,32,66
440 DATA 246,144,121,76,4,247,74,176
450 DATA 3,76,19,247,32,208,247,176
460 DATA 3,76,19,247,32,238,248,176
470 DATA 104,32,175,245,165,183,240,9
480 DATA 32,234,247,144,11,240,90,176
490 DATA 218,32,44,247,240,83,176,211
500 DATA 165,144,41,16,56,208,74,224
510 DATA 1,240,17,224,3,208,221,160
520 DATA 1,177,178,133,195,200,177,178
530 DATA 133,196,176,4,165,185,208,239
540 DATA 160,3,177,178,160,1,241,178
550 DATA 170,160,4,177,178,160,2,241
560 DATA 178,168,24,138,101,195,133,174
570 DATA 152,101,196,133,175,165,195,133
580 DATA 193,165,196,133,194,32,210,245
590 DATA 96
```

Questa routine - per C64 - chiede il nome del file da "leggere" (se non si conosce il nome oppure il nome non esiste, basta premere "RETURN") e quindi visualizza l'indirizzo iniziale e finale di qualsiasi programma registrato su cassetta. I numeri contenuti nelle istruzioni DATA costituiscono una routine in linguaggio macchina, ottenuta modificando opportunamente la LOAD del sistema operativo del C64. Una volta in possesso di questi dati fondamentali sarà semplice caricare il programma tramite il "Supermonitor" e quindi trasferirlo su una cassetta di lavoro. Da quest'ultima, salvo la presenza di altre protezioni, potrà tranquillamente essere ricaricato e salvato con il "Superturbo", quindi mandato in esecuzione con la SYS annotata precedentemente.

Un cursore personalizzato

Non sono mai riuscito a trovare, per il mio C64, un programma che modifichi il cursore - ad esempio in un cerchio od una linea - e che permetta di digitare normalmente qualunque programma con il cursore modificato. Potete, per favore, pubblicarlo su SUPERCOMMODORE?

Davide Lamperti

R. Il cursore altro non è che un carattere di spazio in reverse. Poiché i caratteri in reverse, all'interno del set, sono contraddistinti da codici uguali a quelli dei caratteri normali sommati a 128, dal momento che lo spazio ha il codice 32, il cursore avrà il codice 32+128=160. Sarà quindi sufficiente ridefinire - come un qualunque altro carattere - lo spazio in reverse. Il programma che segue, per il C64, ridefinisce il cursore come una lineetta. Modificando gli otto valori della linea DATA è possibile fargli assumere qualsiasi altra forma.

Il programma, una volta mandato in esecuzione, passerà a leggere immediatamente i caratteri in RAM, prima ancora di iniziare a ridefinirli. Lo schema si riempirà quindi di strani simboli, corrispondenti al contenuto casuale della RAM in cui andremo a ricopiare il set.

Immediatamente inizierà il trasferimento dei caratteri da ROM a RAM, con la loro visualizzazione progressiva sullo schermo; si noti che quando viene ricopiato il 33esimo carattere (i codici partono da 0), che è lo spazio, la parte di schermo non occupata dal set si pulisce istantaneamente, essendo appunto piena di spazi.

Dopo aver trasferito tutto il set da ROM a RAM, si potrà assistere alla ridefinizione dello spazio in reverse, cioè del cursore. Il programma chiederà quindi se si intende utilizzare anche il set minuscolo: quest'ultimo a premere i tasti SHIFT e Commodore per passare al minuscolo, prima di rispondere alla domanda del computer. Un bel caos, vero? Ritornate quindi sul maiuscolo, premendo nuovamente SHIFT e Commodore e, se lo desiderate, continuate con il passare il RAM anche il set minuscolo. Si tenga comunque presente che lo "sfizio" di avere il cursore ridefinito implica - a meno di ricorrere a particolari artifici giocando sui banchi di memoria - la perdita per i programmi in BASIC della memoria a partire dalla locazione 12288. Rimangono liberi quindi solo circa 10 Kbyte. Volendo evitare questo spreco di memoria, occorre fare in modo che il chip VIC-II "veda" un altro banco di memoria da 16 Kbyte, spostando anche la mappa video in un'altra posizione. Il trasferire entrambi i set di caratteri in RAM "mangia" comunque come minimo circa 4 Kbyte di memoria BASIC se si utilizzi il banco 1.

```
10 POKE52,48:POKE56,48:CLR: REM
RISERVA MEMORIA PER IL SET DI
CARATTERI
20 POKE53280,0:POKE53281,0:PRINT
"[CLR]{WHT}":S=1024:PRINTCHR$
(142):REM SET MAIUSCOLO
30 POKE53272,(PEEK(53272)AND240)
+12:REM LEGGE I CARATTERI IN
RAM
40 FORK=0TO255:POKE53280,K:I=I+1:N
EXT:REM VISUALIZZA IL SET
50 POKE56334,PEEK(56334)AND254:R
EM DISATTIVA IRQ TASTIERA
60 POKE1,PEEK(1)AND251:REM DISA
TTIVA L' I/O
70 REM COPIA SET MAIUSCOLO DA RO
M A RAM
80 FORI=0TO2047:POKEI+12288,PEEK
(I+53248):NEXT
90 POKE1,PEEK(1)OR4:REM RIATTIVA
L' I/O
100 POKE56334,PEEK(56334)OR1:REM
RIATTIVA IRQ TASTIERA
110 REM CURSORE = COD. 160
120 REM RIDEFINISCE IL CURSORE N
EL SET MAIUSCOLO
130 FORI=0TO7:READD:POKE(I+12288
+160*8),D:NEXT
140 INPUT"[HOME]{ 10 GIU' }USI AN
CHE IL SET MINUSCOLO (S/N)";
A$
150 IF A$<>"S"THENEND
160 PRINT"[CLR]":PRINTCHR$(14);:
I=0:FORK=0TO255:POKE53280,K:I=
I+1:NEXT:REM VISUALIZZA
170 POKE56334,PEEK(56334)AND254:
REM DISATTIVA IRQ TASTIERA
180 POKE1,PEEK(1)AND251:REM DISA
TTIVA L' I/O
```



```

190 REM COPIA SET MINUSCOLO DA R
    OM A RAM
200 FORI=0TO2047:POKEI+14336,PEE
    K(I+55296):NEXT
210 POKEI,PEEK(1)OR4:REM RIATTIV
    A L' I/O
220 POKE56334,PEEK(56334)OR1:REM
    RIATTIVA IRQ TASTIERA
230 REM CURSORE = COD. 160
240 REM RIDEFINISCE IL CURSORE N
    EL SET MINUSCOLO
250 RESTORE:FORI=0TO7:READD:POKE
    (I+14336+160*8),D:NEXT
260 PRINT"(HOME){ 12 GIU' }"
270 DATA0,0,0,0,0,0,0,255

```

Concatenamento di programmi BASIC

Sono in possesso di un C64 e mi rivolgo a voi per un problema apparentemente banale, ma che non sono ancora riuscito a risolvere. Utilizzando il comando LOAD all'interno di un programma BASIC, per caricare e lanciarne un altro, ho avuto degli spiacevoli inconvenienti: ho infatti notato che se il programma da caricare è più lungo di quello residente nella memoria del computer, esso viene caricato solo parzialmente, con righe modificate e, se lanciato, il computer impazzisce, segnalando gli errori più svariati in righe inesistenti, fornendo un listato incomprendibile, etc.

Ho anche notato che, se il programma da caricare è "più o meno" delle stesse dimensioni di quello in memoria (stesso numero di blocchi occupati nel dischetto, per intenderci), esso viene caricato ed eseguito correttamente, ma il contenuto di alcune variabili (numeriche ed alfanumeriche) viene modificato o cancellato: notare che i programmi in origine a tutta la questione sono scritti interamente in linguaggio BASIC e che le uniche POKE utilizzate sono quelle per la gestione del colore del video e per la cancellazione del buffer della tastiera prima di un'istruzione GET. La riga da me utilizzata per eseguire l'operazione è la canonica LOAD"nome progr.", 8.

P. Fazzari

R. Quello dell'"Overlay", cioè del caricamento di un programma BASIC da parte di un altro programma BASIC, è un argomento spesso trattato superficialmente da manuali e guide di riferimento. In generale, un overlay permette di caricare un nuovo programma nello spazio usato dal precedente programma, mantenendo tutti i valori delle variabili uguali a quelli assunti nel programma precedente, con una limitazione: il programma caricato deve essere di dimensioni uguali o inferiori a quelle del programma caricatore.

Dal momento che l'istruzione LOAD carica a partire dall'inizio della memoria BASIC, il secondo programma sovrascriverà il primo. Le variabili vengono immagazzinate subito dopo la fine del programma e quindi, se il secondo programma risulta più lungo del primo, esse andranno perdute.

Riassumendo, si possono verificare i seguenti casi (chiamiamo per semplicità programma 1 il "caricatore" e programma 2 il "caricato").

1. Il programma 1 è più lungo del programma 2: in questo caso verranno passate al programma 2 tutte le variabili numeriche utilizzate dal programma 1, mentre le variabili di stringa verranno passate solo se dinamiche. Per fare in modo che tutte le variabili di stringa vengano trattate dal computer come dinamiche è sufficiente aggiungerle in coda una stringa nulla: invece di A\$="PIPO", occorrerà usare la forma A\$="PIPO"+". In questo modo le variabili di stringa verranno immagazzinate nella parte alta della memoria.

2. Il programma 2 è più lungo del programma 1: in questo caso andranno inevitabilmente perdute tutte le variabili utilizzate dal programma 1, ed il programma 2 sarà caricato solo in parte, per una lunghezza uguale al programma 1.

Spieghiamo meglio: quando si carica da disco il programma 1, i puntatori del BASIC si regolano sulla lunghezza di questo programma. L'istruzione LOAD di overlay avvia il caricamento del programma 2 più lungo, ma il BASIC crede ancora che la lunghezza del programma sia quella del primo. Il programma 2 verrà quindi caricato solo in parte, e probabilmente con l'ultima linea caricata in forma indecifrabile.

Un modo per ovviare a questi inconvenienti, e passare ugualmente i valori delle variabili ad un programma più lungo, può essere l'utilizzo del buffer di tastiera per riposizionare i puntatori del BASIC sugli stessi valori che si hanno dopo l'accensione del computer, e procedere - sempre con l'utilizzo del buffer di tastiera - al caricamento del secondo programma. Per "passare" il contenuto delle variabili al secondo programma, un'idea può essere quella di sfruttare l'area di memoria da 4 Kbyte che parte dalla locazione 49152: il contenuto di una variabile del primo programma - poniamo il caso forse più complicato di una variabile di stringa A\$="PIPO" - può essere immagazzinato in queste locazioni con un'istruzione del tipo:

```

FORI=1TOLEN(A$):POKE49151+I,
ASCIA(I,I):NEXT:POKE49151+1,0

```

L'inserimento del valore 0 alla fine della stringa serve come indicatore di fine stringa. Il secondo programma potrà ricaricare la variabile di stringa A\$ con il suo contenuto tramite un'istruzione del tipo:

```

10      A=ASC      (PEEK(49152+I):
IFA<>0THEN A$=A$+CHR$(A)&I:GOTO10

```

Naturalmente questo sistema richiede che i due (o più) programmi vengano scritti appositamente con il proposito del passaggio di dati da un programma all'altro. I metodi per

questo passaggio di dati sono numerosi e, soprattutto se si dispone del disk drive, sembra molto più pratico l'utilizzo di un file sequenziale con il contenuto di tutte le variabili.

Come ridefinire i caratteri sul C16

Ho appena acquistato, dopo aver posseduto per anni un VIC 20, il nuovo C16 e mi trovo in gran difficoltà nel riuscire a ridefinire i caratteri, in quanto il manuale fa solo cenno a questa possibilità, senza fornire alcuna indicazione su come realizzarla. Potete aiutarmi?

Giovanni Spadari

R. Inutile ripetere l'inadeguatezza dei manuali forniti dalla Commodore per chi intenda cimentarsi nella programmazione... Quindi solidarizziamo con il lettore e ci affrettiamo a dare spiegazioni sui caratteri ridefiniti nel C16, problema che anche noi abbiamo dovuto risolvere con nottate insonni durante la stesura del "Libro dei giochi per Commodore 16 e Plus/4", da poco uscito.

Il set di caratteri risiede nel C16 e Plus/4 a partire dalla locazione 53248, ovviamente in ROM. Il guaio è che una PEEK in queste locazioni non fornisce il valore contenuto nella ROM, bensì quello contenuto nella RAM "sotto" la ROM, che però... non c'è!

Il sistema migliore per ricopiare il set di caratteri in RAM, ed è quello che abbiamo scelto per il suddetto libro, è sfruttare le routine in linguaggio macchina già disponibili nel computer attraverso il monitor. Si ottiene un trasferimento sicuro di tutto il set con una velocità impressionante.

L'operazione viene eseguita con la tecnica dell'utilizzo del buffer di tastiera, ed è molto semplice. Ecco un esempio per trasferire in RAM il set di caratteri a partire dalla locazione 14336:

```

10 (CLR)ND(6 GIU'):(2 SIN)T D000 D7FF
   3400(2 GIU'):(16 SIN)X(2 GIU'):(SIN)GOTO
   40
20 POKE239,5:POKE1319,19:POKE1320,13:POKE1321,
   13:POKE1322,13:POKE1323,13
30 END
40 POKE5298,PEEK(65298)AND251
50 POKE5299,56
60 POKE52,55:POKE56,55
70 ...
80 ...

```

La linea 10 stampa sul video le istruzioni necessarie per entrare in monitor, operare il trasferimento dei caratteri e ritornare al BASIC. La linea 20 pone nel buffer di tastiera un codice di HOME e quattro codici di RETURN, che verranno quindi stampati sopra le istruzioni visualizzate dalla linea 10, provocando la loro esecuzione.

A questo punto il set di caratteri è stato copiato in RAM, ma bisogna dire al computer di iniziare a "vederlo" in quelle locazioni. A ciò provvedono le linee 40 e 50, mentre la 60 provvede a proteggere il nuovo set da sovrascritture da parte del BASIC.

Nelle linee seguenti si potranno quindi ridefinire i caratteri con il solito sistema delle READ, POKE e DATA.



A che serve l'home computer?

di M. Cristuili Grizzi

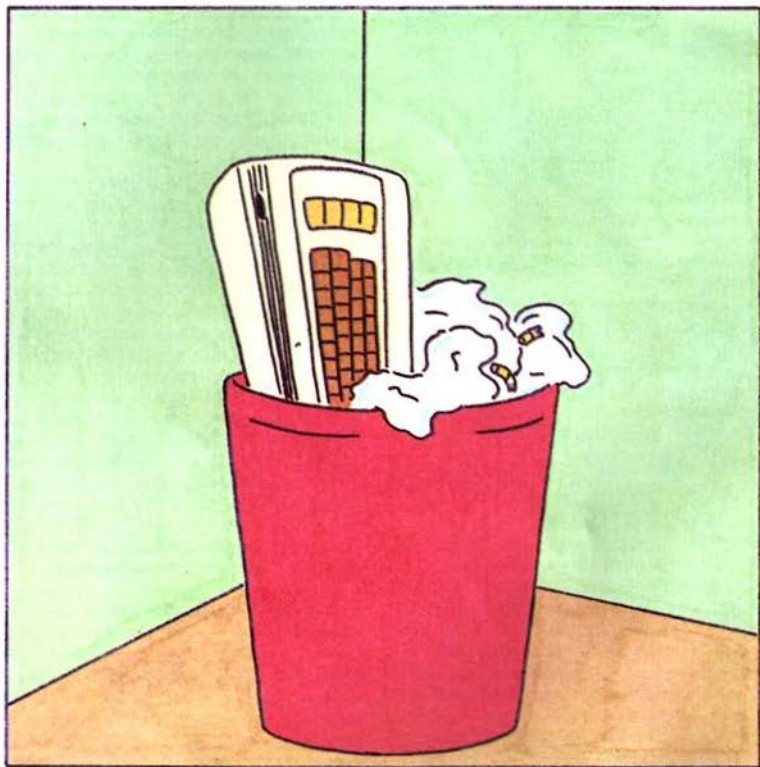
Possedere un home computer, o meglio ancora un personal, è un fenomeno di moda e rappresenta quasi uno status symbol "intellettuale" per molte persone.

La pubblicità delle case costruttrici di computer e l'informazione di massa spingono un numero sempre maggiore di persone all'acquisto della magica macchinetta, promettendone una possibilità di utilizzo costruttiva ed insieme divertente. Per contro, stiamo già assistendo al "rigetto" del computer da parte di chi se ne è presto stufato, per una serie di motivi che vanno dalle difficoltà incontrate nella programmazione al non sapere come usare la propria macchina. In realtà, la domanda che molti si pongono oggi come oggi è: "A che cosa serve un home computer?"

Non è facile dare una risposta di carattere generale a questa domanda, in quanto i termini del problema sono soggetti ad ampie variazioni, a seconda di molti fattori, quali il livello culturale dell'utente, il suo campo di attività lavorativa, il tipo di hobby che coltiva e l'inclinazione del suo carattere.

Si può cercare tuttavia di analizzare il rapporto computer/utente, per tentare almeno di attribuirgli un significato positivo o negativo in alcuni casi generici. Nella gran parte dei casi il computer viene acquistato senza che ne venga sentita una necessità reale, ma solo dal momento che lo possiedono gli amici, per sentirsi "al passo con i tempi", o perché "vi si è spinti dalla campagna pubblicitaria di massa, sia evidente che nascosta".

Un numero quasi pari di persone - novelli acquirenti di computer - non possiede inoltre una cultura logico/matematica sufficiente per programmare personalmente la nuova macchina, o



addirittura per essere in grado di copiare o "far girare" correttamente programmi commerciali o di dominio pubblico. L'effetto frustrante provocato da questa situazione non può che sortire un duplice risultato: da una parte l'abbandono di qualsiasi velleità di approfondimento dell'uso del computer, per passare all'utilizzo completamente pas-

sivo di videogiochi, e dall'altra l'abbandono susseguente e totale del computer, che viene a poco a poco - e silenziosamente per non sfigurare davanti ad amici ai quali ha arriso un maggiore successo - lasciato nel "dimenticatoio".

Data la situazione attuale di sviluppo della piccola informatica in Italia, non è

difficile affermare che il possesso dell'home computer si giustifica generalmente come puro hobby, essendo molto particolari le situazioni in cui esso possa risultare effettivamente utile, anche per una semplice gestione casalinga.

Per fare alcuni esempi, il word processing è senz'altro utilissimo e di gran comodità, ma che senso può avere il suo uso da parte dell'italiano medio che scrive tre lettere private all'anno?

I programmi cosiddetti di gestione casalinga, che si riducono quasi sempre a piccoli database per la rubrica del telefono, il conto in banca, o le ricette di cucina, sono pur sempre battuti in praticità e comodità di utilizzo dai buoni e vecchi libri od agende, da secoli usati con successo: se, infatti, per decidere quale delicatezza cucinare - o per cercare un numero di telefono -, dobbiamo accendere il computer, attendere che venga caricato - quasi sempre da nastro, con tempi lunghissimi - il database, attendere quindi che esso carichi i dati precedentemente inseriti, e solo allora poter avere visualizzata sullo schermo la ricetta od il numero di telefono che ci interessa, vediamo facilmente la superiorità - in tempi medi di accesso a un dato - del vecchio libro o agenda o, più "modernamente", supporto cartaceo.

Si potrebbe obiettare che la procedura sopra descritta è necessaria una volta sola: è possibile caricare il database della rubrica telefonica al mattino, e lasciare il computer acceso fino a sera per poter così ricercare immediatamente un

numero. In questo caso, tuttavia, ci si troverebbe ad aver acquistato non un computer, ma solo un'agenda telefonica elettronica al prezzo di un computer: non si può certo dire di aver compiuto un buon investimento!

Per questi motivi, chi acquista il computer pensando di utilizzarlo per scopi "pratici" deve per prima cosa valutare quali siano le proprie reali esigenze: nella stragrande maggioranza dei casi la famiglia italiana non ha bisogno del computer per "arrivare al 27", o peggio ancora per archiviare informazioni, quando il solo dieci per cento scarso della popolazione acquista quotidianamente un giornale.

Il computer, per sua natura, può essere uno strumento molto utile, ma per poterlo realmente essere necessita - sembra un paradosso, ma molti tuttora non lo capiscono - di un utente che abbia un reale necessità di utilizzarlo.

Diverso è il discorso se il computer viene considerato semplicemente come un hobby: è più "sincero" il rapporto con la macchina, si è in grado - senza aver fretta - di sviluppare, seguendo le pubblicazioni ed i libri specializzati, quella cultura o "forma mentis" logico/matematica tale da permettere di programmare la macchina senza problemi. Grandi sono le soddisfazioni che il computer può offrire quando lo si programmi da sé: ciò che vediamo scorrere sul video è una vera "creazione" del programmatore, personale e scaturita direttamente dalla sua mente. Questo tipo di utilizzo può "aprire la mente", una

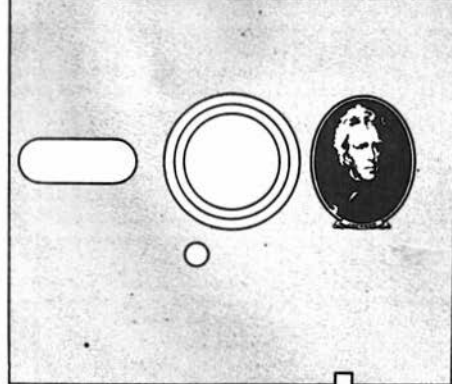
volta lo si diceva dello studio del latino, richiamando quell'elasticità di ragionamento magari un po' arrugginita nelle persone di mezza età, oppure sviluppandola nei ragazzini.

Al contrario, soprattutto se l'utilizzatore è un bambino in tenera età, il computer può diventare uno strumento foriero di alienazioni e stupidità, se usato esclusivamente per passare ore ed ore davanti ad un videogioco: si svilupperanno certamente i riflessi, ma la mente ne risentirà, in mancanza di esercizio costruttivo.

I termini di questo discorso sono destinati a cambiare nell'arco di breve tempo: quando anche in Italia saranno disponibili linee telefoniche affidabili per la trasmissione dati tra computer e computer, finalmente quest'ultimo potrà essere utilizzato in maniera proficua nella vita di tutti i giorni: sarà infatti possibile acquisire una gran quantità di informazioni di qualsiasi genere in brevissimo tempo, fare acquisti o prenotazioni direttamente da casa, spedire e ricevere lettere nello spazio di pochi secondi a migliaia di chilometri di distanza, e tutta una serie di altri servizi che al momento possono ancora apparire "da fantascienza".

Ben venga quindi il computer in tutte le famiglie, facendo però attenzione a non aspettarsi da esso ciò che non è in grado di offrire, e facendo altresì un buon esame di coscienza sulla propria preparazione ad utilizzarlo.

LIBRERIA JACKSON



**A Milano,
in via Mascheroni 14.
La prima software
libreria italiana**

**A Milano, in via Mascheroni 14.
Tel. 02-437385**

**Vieni a trovarci:
ti aspettiamo.**

Un tempo si andava in libreria per il gusto della scoperta, per il piacere di esser informati sulle novità. Per incontrarsi, discutere, chiedere un consiglio al libraio-amico. Tutto questo è ancora possibile, per un prodotto assolutamente nuovo: libri e riviste di informatica italiani ed esteri, software, giochi.

Dove? Alla Libreria JACKSON.
La prima software - libreria italiana.

E' IN EDICOLA

HOBBY ELETTRONICA 1

Mensile di elettronica pratica e hardware per microcomputer



GRUPPO
EDITORIALE
JACKSON

L. 1.000

MAGGIO 85

Nuova edizione della rivista

Speciale
**ANTIFURTO
DIGITALE**

30W E PIU'
D' USCITA
UNITA PORTATILE
PER CHITARRA

COMPUTERJOYSTICK
ANALOGICO

**MOLTO PIU'
DI UNA RIVISTA
DI ELETTRONICA**



GRUPPO
EDITORIALE
JACKSON

...then



ABC del Linguaggio macchina: parte quarta

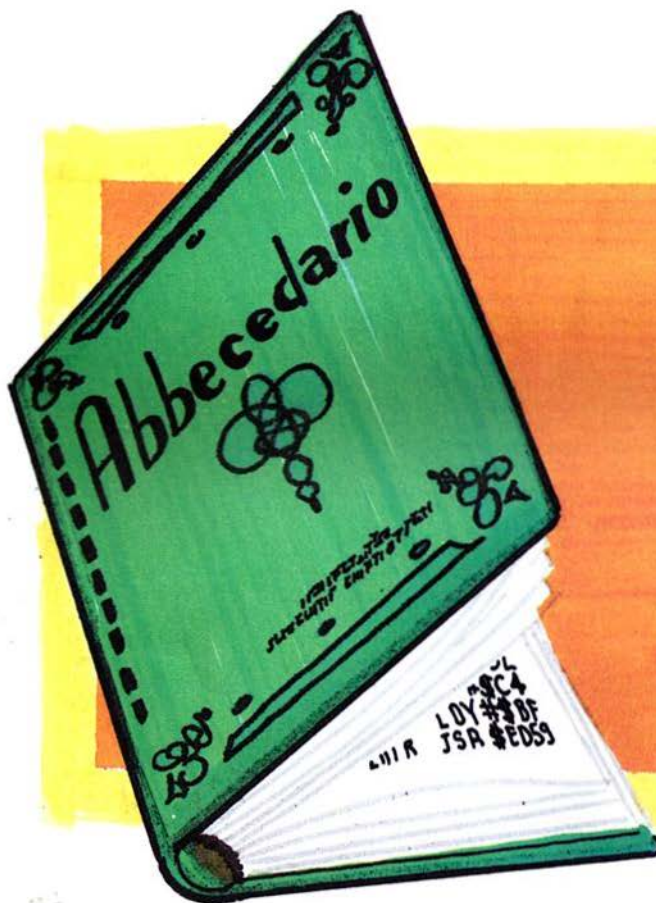
di U. G. Barzaghi

In questo quarto, ed ultimo, appuntamento con la nostra breve rassegna introduttiva sugli aspetti essenziali del linguaggio macchina tratteremo un argomento di estremo interesse: come inserire brevi programmi "personalizzati" nella routine di gestione delle interruzioni del calcolatore.

Vediamo, innanzi tutto, che cos'è una interruzione (in inglese "interrupt").

Una "interruzione" è, semplicemente, un segnale emesso da un componente del vostro calcolatore che ordina al microprocessore, che ne regola il funzionamento, di interrompere qualsiasi funzione stia svolgendo (a meno che questa abbia una maggiore priorità) per eseguire determinati compiti, ben precisati e già noti, e quindi ritornare al punto in cui si era interrotto.

Tipicamente, l'operazione svolta dal microprocessore sarà rappresentata dall'esecuzione del programma interprete BASIC (non diverso da qualsiasi altro programma in linguaggio macchina). All'arrivo di una interruzione, generata ad esempio da uno dei temporizzatori del chip CIA di ingresso/uscita, il sistema operativo sospende l'esecuzione dell'interprete BASIC e cede il controllo ad un altro programma in linguaggio macchina, il cui indirizzo è contenuto nella coppia di locazioni 788-789, il cui compito principale consiste nell'aggiornare gli orologi interni del calcolatore e scandire la tastiera, per rilevare l'eventuale pressione dei tasti. Questo genere di interruzione, generata, come detto, da uno dei temporizzatori del chip CIA, avviene con periodo pari ad un sessantesimo di secondo; la frequenza degli orologi interni del calcolatore è legata alla frequenza di alimentazione e pari, appunto, ad una oscillazione completa ogni sessantesimo di secondo, o "jiffy", come può essere rilevato chiedendo che venga visualizzato il contenu-



to della variabile TI, pari al numero di "jiffy" trascorsi dall'accensione del calcolatore o dall'azzeramento della variabile TI\$.

Questo, però, non è il solo tipo di interruzione che è possibile generare internamente al vostro calcolatore. Una interruzione è rappresentata anche dalla pressione del tasto RUN/STOP, che sospende l'esecuzione di un programma BASIC. In questo caso il compito di riprendere il programma interrotto è la-

sciato all'operatore, che può farlo battendo CONT una volta che abbia ottenuto le informazioni desiderate. È addirittura possibile alterare il contenuto di alcune variabili senza per questo impedire l'eventuale prosecuzione del programma; bisogna fare, però, attenzione a non alterare il contenuto della "pila", richiamando in maniera scorretta sottoprogrammi od interferendo con eventuali cicli FOR...NEXT lasciati aperti dalla nostra interruzione da tastiera.

Listato del programma 1 Cornice & Schermo

C000 SEI	; disabilita le interruzioni
C001 LDA \$0314	; carica in accumulatore il contenuto della locazione 788, contenente il byte meno significativo dell'indirizzo della prima istruzione del sottoprogramma di risposta alle interruzioni
C004 STA \$C020	; il contenuto dell'accumulatore viene trasferito in corrispondenza del byte meno significativo del campo indirizzo dell'istruzione di salto incondizionato che chiude il listato del programma e lo ricollega alla routine di gestione delle interruzioni
C007 LDA \$0315	; viene letto il contenuto della locazione 789 = byte più significativo dell'indirizzo suddetto
C00A STA \$C021	; il byte in questione viene trasferito in corrispondenza del byte più significativo del campo indirizzo della istruzione di JMP di cui sopra
C00D LDA # \$19	; viene caricato in accumulatore il byte meno significativo dell'indirizzo cui corrisponde la prima istruzione del programma che desideriamo premettere alla routine di gestione delle interruzioni
C00F STA \$0314	; il contenuto dell'accumulatore viene trasferito nella prima delle due locazioni di memoria che indicano l'inizio della routine di servizio
C012 LDA # \$C0	; acquisizione del byte più significativo dell'indirizzo di cui sopra
C014 STA \$0315	; il byte viene trasferito nella locazione 789 (byte più significativo della coppia di puntatori)
C017 CLI	; le interruzioni vengono riabilitate
C018 RTS	; rientro in ambiente BASIC
C019 LDA \$D021	; questa istruzione costituisce, insieme alle due che la seguono, il vero e proprio programma aggiunto alla routine di servizio delle interruzioni. Viene letto il contenuto della locazione avente indirizzo decimale 53281 (registro colore di fondo dello schermo)
C01C STA \$D020	; il valore letto dall'istruzione precedente viene trasferito in 53280 (registro colore della cornice)
C01F JMP \$EAB1	; si richiama il sottoprogramma originale di gestione delle interruzioni.

fig. A

Listato del programma 2- 16 Sprites

C000 SEI	; disabilita le interruzioni
C001 LDA \$0314	; viene letto il contenuto della locazione 788 (byte meno significativo della coppia di puntatori alla routine di risposta alle interruzioni)
C004 STA \$C0B1	; il byte letto dall'istruzione precedente viene trasferito in corrispondenza del byte meno significativo del campo indirizzo dell'istruzione di salto incondizionato (JMP), che chiude la routine
C007 LDA \$0315	; analoga alla istruzione di C001 per quanto riguarda il byte più significativo (locazione 789)
C00A STA \$C0B2	; analoga alla C004 per il byte più significativo
C00D LDA # \$28	; viene caricato in accumulatore (usando il formato immediato) il byte meno significativo dell'indirizzo della prima istruzione del programma in linguaggio macchina che si desidera aggiungere alla routine di risposta alle interruzioni
C00F STA \$0314	; il contenuto dell'accumulatore viene trasferito nel byte meno significativo della coppia di puntatori 788/789
C012 LDA # \$C0	; lettura del byte più significativo dell'indirizzo in questione
C014 STA \$0315	; il byte letto dall'istruzione precedente viene trasferito in 789
C017 LDA # \$01	; viene eguagliato a 1 l'accumulatore
C019 STA \$D01A	; il contenuto dell'accumulatore viene trasferito nella Interrupt Request Mask (maschera delle interruzioni), settando il primo bit
C01C LDA # \$00	; viene azzerato l'accumulatore

Se, oltre al tasto RUN/STOP, viene premuto contemporaneamente anche il tasto RESTORE, l'interruzione provocata ha effetti permanenti, la routine di gestione delle interruzioni cancella lo schermo, azzerando il contenuto di tutte le variabili utilizzate e riporta il calcolatore allo stato in cui si trovava prima dell'esecuzione di un programma senza però alterare il listato di eventuali programmi BASIC in memoria.

Oltre alle interruzioni hardware provenienti dai temporizzatori del chip CIA e dalla tastiera, vengono generati "interrupt" anche dal chip VIC-II, dedicato alla gestione della grafica, al verificarsi di determinate combinazioni connesse all'aspetto grafico del vostro calcolatore. La principale di queste interruzioni viene chiamata "raster interrupt" o "interruzione del raggio di scansione". In questo articolo sfrutteremo sia il primo che quest'ultimo tipo di interruzione, per far svolgere al calcolatore dei compiti che richiedono l'impiego di programmi in linguaggio macchina semplici e brevi (quanto minore è il loro tempo di esecuzione, tanto migliori saranno le prestazioni offerte) e la cui esecuzione deve avere frequenza molto alta.

Il modo più semplice e banale di sfruttare la routine di gestione delle interruzioni del vostro calcolatore consiste nell'assegnarle tutti quei compiti che richiedono una frequenza di esecuzione tanto alta da essere improponibile l'idea di svolgerli tramite un normale sottoprogramma BASIC. Se, ad esempio, desiderate visualizzare sullo schermo un cronometro al decimo ed al centesimo di secondo, per cronometrare lo svolgimento di un programma, e non vi accontentate di veder scorrere con discreta "uniformità" le cifre dei minuti e dei secondi, ma pretendereste che anche quelle dei decimi o dei centesimi scorressero senza apparente soluzione di continuità, l'unica soluzione consisterebbe nell'aggiornare le cifre del cronometro tramite una opportuna routine in linguaggio macchina "innestata" sul sottoprogramma di servizio delle interruzioni.

Per illustrare questo primo aspetto dell'argomento trattato ho preferito scegliere un esempio molto più semplice e di immediata comprensione, anche per coloro che non hanno eccessiva dimestichezza con il linguaggio macchina.

L'aspetto fondamentale del problema non è rappresentato, infatti, dalla funzione svolta dal sottoprogramma, bensì dalla tecnica utilizzata per inserire il programma internamente alla routine di gestione delle interruzioni.

Lo schermo del C64 è, come noto, suddiviso in due grandi aree: un'area centrale "visibile" (sul cui contenuto, cioè, il calcolatore può liberamente influire) ed una "cornice", sulla quale l'utente può agire solo cambiandone il colore (inserendo un valore compreso tra 0 e 15, tramite una istruzione POKE, nella locazione di indirizzo 53280). Analoga-

mente, è possibile settare il colore di fondo dell'area dello schermo visibile tramite la locazione 53281.

Personalmente non gradisco affatto l'aspetto estetico della cornice (un rettangolo, immagino, dell'era pionieristica del personal: ai miei tempi gli schermi erano monocromatici...) e preferisco quindi uniformarne il colore al colore di fondo in tutti quei casi in cui non abbia alcuna utilità pratica mantenerli distinti. Per farlo, però, in caso di successivi cambiamenti del colore di fondo è necessario settare il contenuto di DUE locazioni. Ho quindi pensato - a puro fine didattico - di realizzare un programma che mi permettesse di inizializzare la sola locazione che controlla il colore di fondo dello schermo (53281).

Per farlo è sufficiente premettere al sottoprogramma di risposta alle interruzioni una routine che legga il contenuto della locazione 53281 e lo trasferisca inalterato nella 53280. Dal momento che la routine suddetta viene eseguita ogni sessantesimo di secondo (sei volte più rapidamente delle capacità di percezione dell'occhio umano), ogni modifica al contenuto della locazione 53281 si ripercuoterà (quasi) immediatamente sulla locazione 53280.

La tecnica che consente di innestare la nostra routine su quella di servizio degli interrupt consiste nel leggere il contenuto della coppia di puntatori 788/789, che contengono l'indirizzo iniziale della routine di risposta, suddiviso nel consueto formato byte meno significativo/byte più significativo, memorizzandolo in corrispondenza del campo indirizzo dell'ultima istruzione della routine che desideriamo far eseguire ad ogni interruzione. Deve trattarsi, naturalmente, di una istruzione di salto incondizionato (JMP), che colleghi il nostro sottoprogramma alla pre-esistente routine di servizio.



C01E STA \$D012

; vengono azzerati gli otto bit meno significativi del cursore del raggio di scansione (Raster Cursor)
; viene caricato il contenuto della locazione 53265
; al contenuto dell'accumulatore viene sommato "logicamente" il valore numerico #127 (\$7F)
; il risultato dell'operazione precedente viene reimmagazzinato in 53265, azzerando così il bit più significativo del registro cursore del raggio di scansione (che occupa l'ottavo bit della locazione 53265), ma lasciando invariati i sette bit meno significativi della locazione (che controllano diverse opzioni grafiche, tra cui il formato "colore di fondo esteso")
; vengono riabilitate le interruzioni
; rientro in ambiente BASIC

C021 LDA \$D011
C024 AND # \$7F

C026 STA \$D011

C029 CLI
C02A RTS

fig. B

N.B. - Questa prima sezione rappresenta la routine di attivazione del nuovo programma di risposta alle interruzioni

e viene mandata in esecuzione tramite una istruzione SYS49152 da programma o da tastiera.

C02B LDA \$D019

; viene letto il contenuto del registro delle richieste di interruzioni

C02E AND # \$01

; al byte letto dall'istruzione precedente viene applicato l'operatore di somma logica avente argomento 1, in modo da azzerare il bit zero della locazione suddetta, contenente il segnale di interruzione di cursore di schermo

C030 BEQ \$C0B0

; se il risultato dell'operazione precedente dà risultato nullo, si salta all'ultima istruzione del programma, che consente di ricollegarsi alla routine originale di risposta alle interruzioni

C032 STA \$D019

; il risultato dell'operazione di somma logica viene, in caso contrario, reimmagazzinato nella locazione da cui è stato prelevato

C035 LDA \$D012

; vengono caricati in accumulatore gli otto bit meno significativi del registro cursore di schermo

C038 BNE \$C06A

; se il contenuto dell'accumulatore è diverso da zero (cioè se il raggio di scansione si trova lungo lo schermo e non all'inizio della pagina-video), si salta alla routine che controlla la seconda metà dello schermo

C03A LDA # \$91

; viene caricato in accumulatore il valore numerico 145, che costituisce la frontiera tra la prima e la seconda parte dello schermo, nel modo in cui è stato suddiviso dal nostro programma

C03C STA \$D012

; il valore suddetto viene trasferito negli otto bit meno significativi del registro cursore di scansione dello schermo

C03F LDX # \$0F

; viene caricato nel registro X il valore numerico 15, che servirà da indice per il ciclo di lettura delle coordinate degli otto sprite di cui, in realtà, il Commodore 64 dispone

C041 LDA \$0340,X

; viene letto il contenuto della locazione avente indirizzo pari a 832+X (dove per X si intende il contenuto del registro X del microprocessore)

C044 STA \$D000,X

; il valore contenuto in accumulatore viene trasferito nella locazione di indirizzo pari a 53248+X, che corrisponde ad una coordinata (ascissa x, se il contenuto del registro X è pari; ordinata y, se X è dispari) dello sprite di indice $Y=INT(X/2)$

C047 DEX

; viene decrementato di una unità il contenuto del registro indice X

C048 BPL \$C041

; se l'operazione precedente ha dato un risultato maggiore o uguale a zero (non negativo), si ricicla a partire dalla istruzione \$C041

C04A JMP \$C0AA

; si salta alla sezione conclusiva del programma, che si occupa di ripristinare il contenuto dei registri alterati dal sottoprogramma aggiunto di risposta alle interruzioni

fig. C

N.B. - Questa seconda sezione del programma rappresenta la prima parte della routine di risposta alle interruzioni

e controlla i registri x e y degli otto sprite reali per la prima parte dello schermo.

C06A LDA # \$00	; viene azzerato l'accumulatore
C06C STA \$D012	; il contenuto dell'accumulatore viene trasferito negli otto bit meno significativi del registro cursore di scansione
C06F LDX # \$0F	; identica alla C03F
C071 LDA \$0350,X	; analoga alla istruzione C041; in questo caso, però, l'indirizzo di base dei registri da cui viene prelevata l'informazione non è 832, ma 848, vale a dire il secondo gruppo di coordinate degli sprite
C074 STA \$D000,X	; il valore letto dall'istruzione precedente viene trasferito in uno dei registri delle coordinate degli otto sprite reali, con le stesse modalità già viste per la sezione precedente
C077 DEX	
C078 BPL \$C071	; se il contenuto del registro indice X non è negativo, si ricicla all'istruzione \$C071
C07A JMP \$C0AA	; ci si ricollega, anche in questo caso, alla routine conclusiva del programma

fig. D

N.B. - Questa terza sezione presenta una notevole simmetria con la precedente e costituisce la seconda parte del sottoprogramma di gestione delle inter-

ruzioni, controllando gli otto sprite virtuali al di sotto della linea di demarcazione posta a metà video

C0AA PLA	; viene estratto dalla pila (stack) il valore superiore; il puntatore alla testa della pila viene automaticamente aggiornato dopo che il valore è stato trasferito in accumulatore
C0AB TAY	; il valore letto viene trasferito nel registro indice Y, ripristinandone il contenuto
C0AC PLA	
C0AD TAX	; il secondo valore prelevato dalla sommità della pila viene trasferito al registro X
C0AE PLA	; viene estratto dallo stack il valore da assegnare all'accumulatore
C0AF RTI	; rientro da Interrupt (RTI, ReTurn from Interrupt). Vengono reimmagazzinati i valori originali del registro di stato (status flag register) e i due byte che compongono il Contatore di Programma (Program Counter, PC)
C0B0 JMP \$EAB1	; ci si ricollega alla routine originale di gestione delle interruzioni; valgono per questa istruzione tutte le considerazioni fatte per l'analoga istruzione conclusiva del Programma 1

fig. E

N.B. - Quest'ultima parte del programma si occupa di ripristinare il contenuto dei registri Y, X e dell'accumulatore,

oltre al registro di stato e al Program Counter, prelevandone il contenuto dallo Stack.

Figura 1 - Disposizione dei registri virtuali e dei corrispondenti registri reali utilizzati dal Programma 4

Sprite	Funzione	Registri reali	Registri virtuali	
			A	B
0	Ascissa x	53248	832	848
0	Ordinata y	53249	833	849
1	Ascissa x	53250	834	850
1	Ordinata y	53251	835	851
2	Ascissa x	53252	836	852
2	Ordinata y	53253	837	853
3	Ascissa x	53254	838	854
3	Ordinata y	53255	839	855
4	Ascissa x	53256	840	856
4	Ordinata y	53257	841	857

Occorre quindi caricare nella coppia di puntatori suddetta l'indirizzo di partenza della nostra routine, suddividendolo nello stesso modo, affinché, all'arrivo di un "interrupt", venga eseguito come programma di gestione il nostro breve sottoprogramma, che provvederà, a sua volta, a richiamare il sottoprogramma di risposta originale.

È indispensabile rilevare due fattori. Il primo è rappresentato dal tempo di esecuzione della routine aggiunta, che deve, ovviamente, essere limitato, affinché non venga rilevato alcun mutamento nel funzionamento apparente del calcolatore. Sarà quindi necessario che tale sottoprogramma sia il più breve possibile ed abbia un funzionamento quanto più semplice e lineare.

Una seconda osservazione è legata all'alterazione del contenuto dei puntatori 788-789. Queste locazioni contengono, come detto, un indirizzo a partire dal quale, all'arrivo di una interruzione, viene eseguita la routine di servizio. Supponiamo ora di aver dato il via alla parte della nostra routine che si occupa di alterare il contenuto dell'indice in questione e che, nel bel mezzo dell'operazione, si verifichi una interruzione di qualche tipo (ad esempio di aggiornamento degli orologi interni). Ovviamente, in risposta all'interruzione verrà richiamata l'istruzione corrispondente all'indirizzo attualmente contenuto nelle locazioni suddette. Nel caso in cui tale indirizzo non corrisponda né alla prima istruzione della vecchia routine di risposta né a quella del sottoprogramma aggiuntivo, il funzionamento del calcolatore sarebbe, inevitabilmente, compromesso.

È perciò necessario, quando si ha a che fare con la gestione delle interruzioni e dei loro puntatori, premurarsi di rendere il funzionamento del microprocessore momentaneamente insensibile alle interruzioni, disattivandole, tramite istruzioni opportune, prima di aggiornare il contenuto dei puntatori e riattivandole ad operazione avvenuta.

Il nostro semplice sottoprogramma sarà quindi composto da due parti: la prima si premurerà di modificare opportunamente gli indirizzi connessi con la risposta alle interruzioni (non è infatti possibile farlo in BASIC, dal momento che questo linguaggio non consente, ovviamente, di agire sulle procedure di risposta agli "interrupt") e verrà eseguita tramite la consueta istruzione SYS.

La seconda parte sarà invece rappresentata dal vero e proprio programma aggiuntivo, che verrà automaticamente richiamato in risposta alle interruzioni. Affinché il programma sia ampiamente utilizzabile, e non interferisca con il funzionamento di eventuali programmi BASIC che se ne servono, porremo il codice macchina che lo compone a partire dalla locazione 49152. Saremo quindi costretti, ancora una volta, a servirci della versione di "Supermonitor" rilocala a partire dall'indirizzo \$2000

(#8192).

Commentiamo brevemente, come di consueto, il listato del programma. La prima istruzione consente di disattivare le interruzioni, ponendo a uno il bit corrispondente (I) del registro di "flag". In questo modo si impedisce che eventuali "interrupt" sospendano l'esecuzione del programma che si occupa di alterare gli indirizzi che puntano alla routine di servizio delle interruzioni, provocando gli inconvenienti sopra citati.

Le quattro istruzioni che seguono leggono il contenuto della coppia di puntatori 788/789 e lo trasferiscono in corrispondenza del campo indirizzo della istruzione di JMP, che chiude il vero e proprio programma aggiuntivo di risposta alle interruzioni. Dal momento che le istruzioni STA in questione fanno riferimento agli indirizzi assoluti destinati a contenere i byte meno e più significativo dell'indirizzo, il programma che ne risulta non è relocabile. Nel caso, cioè, in cui si desiderasse apportare aggiunte o modifiche al programma, sarebbe necessario accertare gli eventuali nuovi indirizzi a cui corrisponderebbe il campo indirizzo della istruzione di salto incondizionato conclusiva.

Le successive quattro istruzioni caricano nella coppia di puntatori 788/789 l'indirizzo della prima istruzione del programma che si desidera far eseguire come risposta alle interruzioni. Anche tali indirizzi, suddivisi nel consueto formato (byte-più-significativo x 256 + byte-meno-significativo), sono caratteristici del programma che si desidera premettere alla routine di gestione, e vanno quindi controllati in caso di modifiche. Le istruzioni CLI (che azzerano il bit "I" del registro di "flag", riabilitando quindi le interruzioni) e RTS, che consente il rientro in BASIC, chiudono la parte di programma che si occupa di modificare i puntatori, in modo da inserire il programma che si desidera far eseguire all'inizio del programma di gestione delle interruzioni e di collegarlo, quindi, alla routine originale.

Questa prima parte, di conseguenza, non subisce modifiche in caso di cambiamento del programma che si desidera inserire, se non per quanto riguarda gli indirizzi più volte citati.

Le tre istruzioni che concludono il nostro breve programma esemplificativo rappresentano il vero e proprio programma aggiuntivo. Le prime due non fanno altro che trasferire in un registro il contenuto dell'altro, in modo da ottenere l'effetto desiderato. Esse sono caratteristiche del nostro esempio e non ci soffermeremo quindi a commentarle oltre.

Più interessante è l'istruzione di salto incondizionato che chiude il programma. Essa è necessaria per ricollegare il programma inserito alla routine originale di servizio delle interruzioni. Non sarebbe necessario, a rigor di logica, specificare un particolare indirizzo, poiché

5	Ascissa x	53258	842	858
5	Ordinata y	53259	843	859
6	Ascissa x	53260	844	860
6	Ordinata y	53261	845	861
7	Ascissa x	53262	846	862
7	Ordinata y	53263	847	863
0	Indice forma	2040	864	872
1	Indice forma	2041	865	873
2	Indice forma	2042	866	874
3	Indice forma	2043	867	875
4	Indice forma	2044	868	876
5	Indice forma	2045	869	877
6	Indice forma	2046	870	878
7	Indice forma	2047	871	879
0	Colore	53287	880	888
1	Colore	53288	881	889
2	Colore	53289	882	890
3	Colore	53290	883	891
4	Colore	53291	884	892
5	Colore	53292	885	893
6	Colore	53293	886	894
7	Colore	53294	887	895

Listato del Programma 4

```

C000 SEI ; disabilita interruzioni
C001 LDA $0314 ; [788] → ACCUM
C004 STA $C084 ; ACCUM → [49284]
C007 LDA $0315 ; [789] → ACCUM
C00A STA $C085 ; ACCUM → [49285]
C00D LDA # $2B ; ACCUM ← 43
C00F STA $0314 ; ACCUM → [788]
C012 LDA # $C0 ; ACCUM ← 192
C014 STA $0315 ; ACCUM → [789]
C017 LDA # $01 ; ACCUM ← 1
C019 STA $D01A ; ACCUM → [53274]
C01C LDA # $00 ; ACCUM ← 0
C01E STA $D012 ; ACCUM → [53266]
C021 LDA $D011 ; [53265] → ACCUM
C024 AND # $7F ; (ACCUM AND 127) → ACCUM
C026 STA $D011 ; ACCUM → [53265]
C029 CLI ; riabilita le interruzioni
C02A RTS ; rientra in BASIC

```

fig. 2

N.B. - Questa prima parte rappresenta la routine di attivazione del programma e coincide sostanzialmente con la prima sezione del Programma 2. Fanno eccezione gli indirizzi (\$C084 e \$C085), i

quali ospitano i byte meno e più significativo del campo indirizzo della istruzione di salto incondizionato che ricollega il nostro programma alla routine originale di risposta alle interruzioni.

```

C02B LDA $D019 ; [53273] → ACCUM
C02E AND # $01 ; (ACCUM AND 1) → ACCUM
C030 BEQ $C083 ; Se il bit Z del registro di flag è settato a uno, salta all'indirizzo specificato ($C083 = # 49283)
C032 STA $D019 ; ACCUM → [53273]
C035 LDA $D012 ; [53266] → ACCUM
C038 BNE $C05D ; Se Z=0, salta all'indirizzo indicato ($C05D = # 49245)
C03A LDA # $91 ; ACCUM ← 145
C03C STA $D012 ; ACCUM → [53266]
C03F LDX # $0F ; X ← 15
C041 LDA $0340,X ; [832+X] → ACCUM

```


C044 STA \$D000,X	: ACCUM → [53248+X]
C047 TXA	: X → ACCUM
C048 CLC	: CARRY ← 0
C049 ROR	: Rotazione verso destra del contenuto dell'accumulatore. Come abbiamo visto nella seconda parte della nostra breve rassegna, ciò equivale a dividere un numero per due, con troncamento del resto.
C04A TAY	: ACCUM → Y
C04B LDA \$0360,Y	: [864+Y] → ACCUM
C04E STA \$07F8,Y	: ACCUM → [2040+Y]
C051 LDA \$0370,Y	: [880+Y] → ACCUM
C054 STA \$D027,Y	: ACCUM → [53287+Y]
C057 DEX	: X-1 → X
C058 BPL \$C041	: Se il bit N è nullo, salta all'istruzione avente indirizzo esadecimale \$C041 (# 49217)
C05A JMP \$C07D	: salto incondizionato all'istruzione di indirizzo \$C07D (# 49277)

fig. 3

N.B. - Questa seconda sezione smista (nella prima parte, da \$C02B a \$C038) il corso del programma verso la sezione opportuna (normale routine di gestione delle interruzioni, routine di controllo della parte alta o della parte bassa dello schermo). La prima parte della seconda sezione presenta delle modifiche di dettaglio rispetto alla analoga sezione del Programma 2 (indirizzi assoluti di salto). La seconda parte (\$C03A-\$C05A) controlla i registri virtuali per la parte alta dello schermo. Le modifiche rilevanti, rispetto alla versione precedente, riguardano, ovviamente, la parte di con-

trollo dei registri aggiunti (\$C047-\$C054). Notate come il contenuto del registro indice X (il cui valore varia da 15 a 0) venga trasferito in accumulatore, fatto ruotare verso destra - dopo aver preventivamente azzerato il bit di riporto (Carry), che viene coinvolto nella rotazione - in modo da dimezzare il valore dell'indice. Il valore così ottenuto viene trasferito nel registro indice Y, che viene utilizzato per le successive operazioni di gestione dei registri virtuali di forma e colore, per i quali il valore dell'indice deve variare da 7 a 0.

il programma di attivazione costituito dalla prima parte del programma si pre-mura di inserire in corrispondenza del campo indirizzo dell'istruzione i valori opportuni, desumendoli dalla coppia di puntatori 788/789. Il valore che abbiamo specificato, però, corrisponde alla posizione della routine di servizio delle interruzioni in condizioni di funzionamento standard, quando, cioè, non si sia provveduto ad alterare i banchi di memoria controllati dal chip VIC-II o la memoria di definizione dello schermo o dei caratteri.

L'operazione di attivazione si rende quindi necessaria, in modo da leggere il contenuto attuale dei puntatori e trasferirlo in corrispondenza dell'istruzione conclusiva del programma.

È quindi possibile "leggere" la routine di servizio originale, utilizzando una delle due versioni del nostro programma assembler (anche la versione originale ha accesso alla gamma di indirizzi indicata), tramite una istruzione di disassemblaggio.

.D EAB1

Una volta terminato l'inserimento del codice macchina in memoria, tramite la versione rilocala del nostro monitor, possiamo salvarne una copia su nastro o disco tramite le istruzioni

S C000 C022 "CORNICE & SCHERMO"

o, rispettivamente

S C000 C022 "CORNICE & SCHERMO" 08

I file così ottenuti potranno essere caricati direttamente nella posizione voluta, leggendo dalla periferica desiderata, specificando il suffisso ".1" dopo il numero caratteristico della periferica stessa.

Per attivare il programma sarà sufficiente utilizzare, anche in formato diretto, l'istruzione SYS49152.

Ciò eseguirà la prima parte del programma, che altera gli indirizzi contenuti nella coppia dei puntatori e nella istruzione di salto incondizionato.

Ora, ogni volta che si verifica una interruzione la coppia 788/789 indicherà LDA \$D021 come prima istruzione della routine di servizio. Il nostro breve programma verrà quindi eseguito ogni volta e provederà a richiamare automaticamente la vera routine di servizio.

Potete provarne gli effetti, inserendo valori numerici compresi tra 0 e 15 nella locazione 53281 tramite una semplice istruzione POKE, anche da tastiera. La cornice assumerà ogni volta lo stesso colore dello schermo, così come era immediatamente avvenuto al momento dell'attivazione del programma, nel caso in cui i due colori fossero differenti. Per disattivare il programma è sufficiente premere contemporaneamente i tasti RUN/STOP e RESTORE, che, provo-

C05D LDA # \$00	ACCUM ← 0
C05F STA \$D012	ACCUM → [53266]
C062 LDX # \$0F	X ← 15
C064 LDA \$0350,X	[848+X] → ACCUM
C067 STA \$D000,X	ACCUM → [53248+X]
C06A TXA	X → ACCUM
C06B CLC	CARRY ← 0
C06C ROR	Rotazione a destra dell'accumulatore
C06D TAY	ACCUM → Y
C06E LDA \$0368,Y	[872+Y] → ACCUM
C071 STA \$07F8,Y	ACCUM → [2040+Y]
C074 LDA \$0378,Y	[888+Y] → ACCUM
C077 STA \$D027,Y	ACCUM → [53287+Y]
C07A DEX	X-1 → X
C07B BPL \$C064	Se il bit N è nullo, si salta alla locazione indicata (\$C064 = # 49252)

fig. 4

N.B. - Questa terza sezione è, anche in questo caso, fortemente simmetrica rispetto alla seconda parte della secon-

da sezione, e provvede al controllo dei registri virtuali per la parte bassa dello schermo.

C07D PLA	ACCUM ← [STACK(S)] e S+1 → S. Dove STACK(S) rappresenta la locazione corrispondente alla sommità della pila, puntata dall'indice S, che viene conseguentemente aggiornato
C07E TAY	ACCUM → Y
C07F PLA	ACCUM ← [STACK(S)] e S+1 → S
C080 TAX	ACCUM → X
C081 PLA	ACCUM ← [STACK(S)] e S+1 → S
C082 RTI	Rientro da interrupt
C083 JMP \$EAB1	Salto incondizionato all'indirizzo indicato

fig. 5

N.B. - Quest'ultima parte del programma realizza il consueto ripristino del contenuto dei registri Y, X e dell'accu-

mulatore, quindi si ricollega alla routine originale di gestione delle interruzioni.

cando un diverso tipo di interruzione, ripristinano i puntatori 788/789 ai valori originali (oltre a riconfigurare altri aspetti del sistema, tra cui i valori iniziali contenuti nei registri dei colori della cornice e dello schermo).

Raster interrupt

Per "raster interrupt" o "interruzione del raggio di scansione" si intende un tipo particolare di interrupt, la cui emissione è legata ad un particolare aspetto della grafica del sistema.

L'immagine riprodotta su di un qualsiasi schermo TV è determinata da un "pennello elettronico" (in pratica un fascio di elettroni opportunamente deviato da un campo elettromagnetico), che percorre rapidamente lo schermo, illuminando determinati punti della superficie visibile del tubo catodico e tracciando, in questo modo, l'immagine. In condizioni di funzionamento normali il televisore riceve l'informazione sotto forma di segnale proveniente dall'antenna, decodifica il segnale e riproduce l'informazione, nel modo suddetto, sullo schermo. La rapida variazione del segnale ricevuto, e l'incapacità dell'occhio umano di percepire variazioni più rapide del decimo di secondo, creano l'illusione della continuità del movimento per le immagini sullo schermo. In realtà, il nostro "pennello elettronico" scandisce lo schermo, partendo dall'angolo superiore sinistro e muovendosi orizzontalmente verso destra, e durante il percorso illumina parti opportune della riga. Una volta raggiunto il margine destro il raggio scende di una riga e ricomincia da sinistra.

Queste evoluzioni del pennello elettronico sono visibili nella ripresa non sincronizzata di uno schermo TV da parte di una cinepresa o di una telecamera, sotto forma di righe orizzontali in movimento apparente lungo lo schermo. Il funzionamento dei monitor e dei televisori a colori non differisce di molto dal precedente (nella maggior parte dei casi i pennelli elettronici sono tre — uno per ciascuno dei colori fondamentali — invece di uno).

Il comportamento del televisore, nel caso in cui serva da monitor, differisce solo perché il segnale viene generato dal chip grafico del computer invece che essere ricevuto tramite una antenna.

Lo schermo del Commodore 64 è composto da 263 righe orizzontali, ma solo 200 di queste sono visibili (le prime 50 e le ultime tredici costituiscono il bordo superiore ed inferiore della cornice). Il più volte citato chip grafico VIC-II dispone di un registro cursore (raster register, in 53266) il cui contenuto indica la riga di schermo nella quale si verificherà la prossima interruzione. L'aspetto più notevole di questo registro è rappresentato dal fatto che, qualora venga inizializzato invece che letto, esso indica la riga di schermo in cui si desidera che la prossima interruzione si verifichi.

Il resto del procedimento non differisce gran che da quanto visto in precedenza. In risposta all'interruzione viene richiamata la solita routine, alla quale è necessario premettere il programma in linguaggio macchina che si desidera far eseguire.

Ciò che è necessario tener presente è che il programma in questione dovrà settare il registro cursore in corrispondenza della prossima riga in cui si desidera che l'interruzione avvenga, prima di richiamare il normale sottoprogramma di risposta alle interruzioni.

Questa tecnica può essere impiegata per ottenere svariati effetti grafici: si può dire che l'unico limite alle possibilità di impiego dei "raster interrupt" è data dalla fantasia dei programmatori. Si può, ad esempio, cambiare il contenuto dei registri di controllo del colore di fondo per creare aree dello schermo aventi colori di fondo differenti. Oppure si può agire sui registri di controllo dei formati grafici, mescolando così i vari formati su di una stessa schermata. Oppure si può cambiare il contenuto dei puntatori alla memoria di schermo (attenzione, però, in questo caso a controllare opportunamente il contenuto della memoria colore) o, se si dispone di più set di caratteri personalizzati, dei puntatori al set di caratteri, mescolando così, su di una stessa pagina video, forme grafiche diverse.

Va fatta una precisazione: le prestazioni del programma saranno tanto migliori quanto esso sarà più rapido. Pertanto è inutile pretendere di suddividere lo schermo in quattro o cinque aree orizzontali, alterando in ognuna il contenuto di una mezza dozzina di registri.

Già suddividere lo schermo in tre aree, aventi formati grafici diversi, rappresenta una prestazione non indifferente. Una simile esempio è dato dall'articolo di S. Leemon "Come utilizzare formati grafici diversi sul C64", già apparso in uno dei numeri precedenti di "Supervic" e riportato in uno dei due volumi

di "C64: suono e grafica", pubblicati dalla J.soft.

Ho preferito, in questo articolo, presentare un esempio originale, corredato da un programma dimostrativo. L'opzione grafica da me scelta è rappresentata, ancora una volta, dagli sprite.

Come è noto, il C64 dispone di otto di queste figure grafiche, rappresentabili in una matrice di 24 punti orizzontali per 21 verticali e di cui è possibile controllare la posizione, il colore, la priorità, rispetto ai caratteri del testo che compaiono sullo schermo, e il formato grafico (normale o multicolore).

Ciò è sufficiente nella maggior parte delle applicazioni; può darsi, però, che a volte si desideri disporre di un maggior numero di questi "folletti" benevoli: il programma in linguaggio macchina da me realizzato consente di raddoppiarne il numero, controllandone separatamente la forma e la posizione, a patto che il movimento di otto di essi si svolga interamente al di sopra di una determinata linea di demarcazione posta sullo schermo, mentre i restanti otto ne rimangono al di sotto, fig. 2.

La tecnica utilizzata dal programma per raddoppiare il numero degli sprite è particolarmente semplice. Tramite il meccanismo delle interruzioni raster, viene creata una suddivisione dello stesso in due zone, separate da una linea di demarcazione prefissata dal programmatore (in questo caso in corrispondenza della 145esima riga orizzontale del video). Ciascuna serie di registri reali, la quale controlla gli effetti che si desiderano implementare, viene sostituita da due serie di altrettanti registri virtuali, una delle quali controlla gli sprite al di sopra della linea di demarcazione, l'altra, ovviamente, al di sotto.

Il meccanismo della duplicazione consiste appunto nel trasferire i dati dall'opportuna serie di registri virtuali nell'unica serie di corrispondenti registri reali, prelevandoli dall'una o dall'altra serie di registri virtuali, in base alla posizione del raster.

Il programma proposto presenta notevoli limitazioni ed ha scopo essenzialmente esemplificativo. Nella sua realizzazione si è tenuto conto di questo fatto, suddividendo il programma in quattro sezioni chiaramente identificabili (da \$C000 a \$C02A, da \$C02B a \$C04C, da \$C06A a \$C07C e da \$C0AA a \$C0B2), separate tra loro in modo da permettere successive evoluzioni del programma, con il controllo di altre serie di registri degli sprite.

In questa prima versione si è scelto di controllare i soli registri delle coordinate degli sprite. Per ragioni di semplicità si è preferito rinunciare ad un controllo della ascissa x oltre la 255-esima posizione (la qual cosa avrebbe comportato una notevole complicazione del progetto, dal momento che tale valore è controllato da un singolo bit del registro 53264 per ciascuno degli sprite, come abbiamo già visto nella seconda parte



di questo mini-corso). Non vengono, inoltre, alterati i valori del colore dei singoli sprite e la rispettiva priorità per le due serie. Ciò significa che, supponendo di aver otto sprite reali di otto colori diversi, avremo uno sprite di un determinato colore al di sopra e uno al di sotto della linea di demarcazione. Analogamente, non viene controllata l'attivazione dei singoli sprite, né la loro forma. Vale a dire, il numero di sprite che compare sullo schermo è controllato dal programma che si serve della routine, ma sempre pari ed equamente ripartito tra le due zone dello schermo (compariranno tanti sprite al di sopra della linea di demarcazione quanti ne compaiono al di sotto).

In pratica, viene controllato il solo contenuto dei registri reali da 53248 (ascissa x dello sprite 0) a 53263 (ordinata y dello sprite 7), in cui viene, di volta in volta, trasferito il contenuto delle due serie di registri virtuali da 832 a 847 e da 848 a 863 (per comodità si è preferito utilizzare il buffer di cassetta per i registri virtuali).

Ovviamente, i valori inseriti in tali registri devono essere congruenti con le limitazioni viste per il programma. Nel caso, cioè, in cui venisse assegnato ad una ordinata y virtuale appartenente alla prima sezione (da 832 a 847; $y < 145$) un valore che travalica la linea di demarcazione, lo sprite corrispondente scomparirebbe, semplicemente, dallo schermo. La qual cosa suggerisce un metodo particolarmente efficace per avere un numero di sprite sullo schermo diverso per le due parti.

Un analogo ragionamento si può fare per qualsiasi registro y virtuale della seconda sezione che contenga un valore inferiore a quello che caratterizza la linea di demarcazione.

Facciamo un esempio pratico. Supponiamo che lo sprite 0 sia attivato (bit 0 del registro 53269 posto a 1), che il suo colore sia giallo (53287 uguale a 7) e che la sua forma sia indicata dal puntatore corrispondente in 2040 (nel caso in cui la pagina di schermo ed il banco di memoria controllato dal chip video VIC-II si trovino nelle condizioni standard).

Le sue coordinate sono controllate dai due registri reali in 53248 (ascissa x) e 53249 (ordinata y), con l'eventuale aggiunta di 256 pixel verso destra nel caso in cui sia attivato il bit 0 del registro 53264. I registri virtuali corrispondenti sono ospitati dalle locazioni 832 e 833, per la prima metà dello schermo, e da 848 e 849 per la seconda.

Il fatto che il bit 0 del registro 53264 non venga controllato significa, semplicemente, che l'ascissa x dei due sprite in cui lo sprite 0 viene sdoppiato sarà o a destra o a sinistra della 255-esima posizione per entrambi.

Per ragioni di rapidità di esecuzione (non dimentichiamo che stiamo operando su delle interruzioni!) il programma non è in grado, inoltre, di svolgere alcun

controllo di correttezza sui valori inseriti, che dovranno quindi soddisfare alle limitazioni suddette.

Vediamo in pratica cosa succede (supponendo che il nostro programma in linguaggio macchina sia stato attivato da una istruzione SYS49152): il programma riceve un raster interrupt in corrispondenza della prima linea dello schermo; accede alla prima sezione della routine aggiunta al sottoprogramma di risposta alle interruzioni e trasferisce nei registri reali 53248 e 53249 il contenuto dei registri ideali della prima serie (832/833). Supponendo che tali registri contengano i valori 200 e 120 (< 145), vedremo apparire uno sprite giallo nel punto di coordinate (200, 120). Il successivo raster interrupt viene generato in 145 (a ciò provvede la prima sezione, iniziando opportunamente il contenuto del registro del cursore di scansione, prima di ricollegarsi alla normale routine di risposta), viene richiamata la sezione che controlla i registri virtuali oltre la 145-esima posizione verticale e questa trasferisce negli stessi registri reali 53248 e 53249 il contenuto della seconda serie di registri virtuali (848/849), quindi fissa il successivo raster interrupt perché si verifichi in corrispondenza della riga 0 e richiama la routine di risposta originale. Supponendo che le coordinate contenute in 848/849 siano uguali a 50 e 200, lo sprite 0 scomparirà in (200, 120) e ne riapparirà uno della stessa forma e colore in (50, 200). Dato, però, che il raggio di scansione dello schermo ha una rapidità molto superiore alle capacità di percezione dell'occhio umano (a patto che il programma da noi aggiunto non risulti troppo lento), ciò che in effetti vedremo saranno DUE sprite gialli, identici in (200, 120) e in (50, 200).

Spero che la spiegazione precedente non vi abbia frastornati. Aggiungerò solo che eventuali richieste di interruzioni provenienti dai temporizzatori verranno servite direttamente dalla routine originale, grazie al test in \$C030 (si potrebbe addirittura pensare di premettere a questa routine il programma 1!).

Chi desiderasse, inoltre, veder comparire uno solo dei due sprite non dovrebbe far altro che cambiare il contenuto del registro y virtuale nel modo su indicato (ponendolo cioè ad un valore minore di 145, se si desidera far scomparire lo sprite inferiore; maggiore o uguale a 145, se non si vuole veder apparire lo sprite superiore).

Per illustrare debitamente le possibilità offerte dal Programma 2 è stato realizzato il Programma dimostrativo 3.

Vedere programma 3.

La versione qui riportata prevede che la routine del Programma 2 venga copiata tramite la versione rilocalizzata di "Supermonitor", a partire da \$C000 nelle locazioni i cui indirizzi appaiono a fianco delle istruzioni, e registrata su disco sotto il nome "16 SPRITES". Il disco su cui viene registrata deve essere inserito

nel drive al momento dell'esecuzione del programma in modo che la routine venga letta e trasferita direttamente nell'area di memoria a lei destinata, tramite la tecnica diffusamente illustrata nella scorsa puntata di "ABC del linguaggio macchina". In alternativa è possibile convertire il Programma 2 in una serie di valori numerici decimali (< 255), che vanno inseriti nelle opportune locazioni di memoria leggendoli da frasi DATA incorporate al Programma 3.

Anche questa tecnica è stata diffusamente spiegata nelle puntate precedenti.

Per avere un'idea delle modifiche si può prendere a modello quanto fatto per il Programma 5, che rappresenta una dimostrazione della versione modificata della routine "16 SPRITES", in modo che, oltre a controllare la posizione degli sprite, se ne alterino anche forma e colore.

Una versione evoluta della routine di raster

Le principali differenze tra le due routine riguardano il solo numero di registri reali controllati. In aggiunta alle coordinate degli sprite (53248-53263), si è scelto di controllare i registri colore degli sprite (53287-53294) e gli indici che puntano alla forma di ciascun sprite (se il chip grafico controlla il primo banco di memoria e la memoria di schermo si trova in 1024-2047, i registri suddetti occupano le locazioni 2040-2047).

Ciò significa che in aggiunta alle due serie di registri virtuali utilizzate dal Programma 2 dovremo disporre di due serie di otto registri colore virtuali e di due serie di otto registri puntatori. Utilizzando le locazioni della memoria tampone del registratore successive a quelle che ospitano i registri virtuali delle coordinate, si è giunti alla seguente configurazione: fig. A.

Il meccanismo del programma non varia sostanzialmente, in quanto a ciascuna sezione è stata semplicemente aggiunta una serie di istruzioni, che consente di controllare i registri ausiliari di forma e colore.

La differenza più evidente è costituita dalla maggior lentezza del programma di risposta alle interruzioni da noi realizzato, il quale provoca così un evidente e spiacevole effetto di lampeggio, che non è comunque possibile eliminare.

Dal momento che questo fatto impedisce ulteriori espansioni del programma, ho preferito compattarne il listato, giustappendendo una all'altra ciascuna delle quattro sezioni da cui è costituito, la qual cosa rende più semplice trasferire, nelle locazioni di memoria prescelte per ospitarla, la traduzione delle istruzioni di codice macchina in valori decimali.

Il programma precedente presenta modifiche tanto semplici alla versione originale da non richiedere neppure un commento dettagliato.

Mi limiterò ad evidenziare il metodo uti-

lizzato per controllare tutti i registri all'interno di un unico ciclo di lettura, che per il registro X va da 15 a 0 (il che permette di trasferire sia le ascisse x che le ordinate y degli sprite dai registri virtuali ai registri reali). Il contenuto del registro suddetto viene quindi trasferito all'accumulatore, ruotato verso destra (ricordiamo come gli operatori di rotazione disponibili per il 6502 agiscano sul solo accumulatore) e quindi trasferito al registro y, dove verrà utilizzato per scorrere le serie dei registri forma e colore virtuali (entrambe composte da otto registri). Supponiamo, in pratica, che X valga 13 (il che equivale a puntare l'ordinata y dello sprite 6). Il suo valore binario corrisponde a

00001101

Azzerando il riporto, e ruotando verso destra, il bit meno significativo cadrà nel Carry stesso, mentre il bit più significativo viene rimpiazzato da uno zero, ottenendo la seguente configurazione

binaria

00000110

cui corrisponde il valore decimale 6. Tale valore, trasferito nel registro indice Y, consente di raggiungere i registri forma e colore corrispondenti appunto al sesto sprite.

Come i più esperti avranno notato, l'operazione precedente viene in realtà compiuta due volte (una volta in corrispondenza al trasferimento dell'ascissa ed una volta per l'ordinata di ciascuno sprite), quindi, in realtà, la seconda iterazione dell'operazione sarebbe superflua. Dal momento però che il suo risultato non provoca alterazione del contenuto dei registri forma e colore, risulta molto più conveniente trascurare questa ripetizione che considerare il caso esplicitamente (a meno di riscrivere interamente la routine, utilizzando indici da 7 a 0, opportunamente "manipolati" per puntare ai registri contenenti le coordinate).

Anche questo secondo programma di

gestione del raster interrupt è corredato da un programma dimostrativo.

In questo caso si è preferito utilizzare la conversione del codice macchina del programma in valori decimali, inseriti in frasi DATA, e letti e trasferiti in memoria al momento dell'esecuzione.

Dal momento che il programma deve anche disporre della forma di 16 sprite differenti (per poter apprezzare il controllo di tale opzione), e che i 64 valori numerici che definiscono la forma di ciascuno sprite devono, anch'essi, essere letti ed inseriti in memoria, il programma risulta un po' lento all'inizio.

Per dimostrare le possibilità offerte tramite una opportuna scelta delle coordinate dei 16 sprite virtuali, ho pensato di far apparire gli sprite sullo schermo uno alla volta, far loro compiere brevemente qualche evoluzione e quindi farli scomparire uno alla volta in ordine inverso. Ragioni di spazio mi impediscono di commentare più diffusamente il listato del programma dimostrativo.

Programma 5

ABC del linguaggio macchina programma 3

```
5 IFSE=0 THEN SE=1:LOAD"16 SPRITES",1,1
:rem 231
10 PRINT"{CLR}{WHT}":POKE53281,0:POKE5328
0,0:V=53248:GOSUB170
:rem 126
20 PRINT"{HOME}{ 11 GIU' }[<L>][< 38 +>]
[<J>]"
:rem 16
30 PRINT"{SU}[<L>][< 38 +>][<J>]"
:rem 212
40 DIMX(16),Y(16):FORI=0TO15:X(I)=SGN(RND
(TI)-.5):Y(I)=-X(I):NEXT
:rem 113
50 FORI=0TO7:POKE832+I*2,INT((RND(I)*160)
/10)*10+90
:rem 89
60 POKE848+I*2,INT((RND(I)*160)/10)*10+90
:rem 176
70 POKE833+I*2,INT((RND(I)*70)/10)*10+50
:rem 119
80 POKE849+I*2,INT((150+RND(I)*100)/10)*1
0
:rem 218
90 POKEV+39+I,I+1:POKE2040+I,255-I:POKEV+
21,255:NEXT
:rem 57
100 SYS49152:REM POKEV+23,255:POKEV+29,
255
:rem 173
110 FORI=0TO15:X=PEEK(832+I*2):Y=PEEK(833
+I*2)
:rem 158
120 IFX=250 THEN X(I)=-1:GOTO140
:rem 18
130 IFX=90 THEN X(I)=+1
:rem 219
140 IF(Y=120 AND I<8) OR (Y>=220 AND I>7) THEN Y(I)
=-1:GOTO160
:rem 224
150 IF(Y=50 AND I<8) OR (Y=150 AND I>7) THEN Y(I)
=+1
:rem 107
160 X=X+X(I)*10:Y=Y+Y(I)*10:POKE(832+I*2)
,X:POKE(833+I*2),Y:NEXT:GOTO110
:rem 212
170 FORK=0TO7:FORJ=0TO63:READB:POKE(255-K
)*64+J,B:NEXT:NEXT:RETURN
:rem 222
180 DATA000,126,000,001,255,128,007,223
:rem 150
190 DATA224,015,159,144,030,063,152,028
:rem 172
```

```
200 DATA063,056,063,063,060,062,126,124
:rem 164
210 DATA126,126,126,126,124,254,120,024
:rem 160
220 DATA254,127,241,254,063,227,252,063
:rem 175
230 DATA135,252,030,031,248,024,127,248
:rem 168
240 DATA009,255,240,007,255,224,001,255
:rem 167
250 DATA128,000,126,000,000,000,000,000
:rem 121
260 DATA000,126,000,001,255,128,007,255
:rem 154
270 DATA224,014,031,144,028,207,152,025
:rem 163
280 DATA207,056,063,031,060,062,126,124
:rem 167
290 DATA124,254,126,121,220,254,120,024
:rem 160
300 DATA254,127,241,254,063,227,252,063
:rem 174
310 DATA135,252,030,031,248,024,127,248
:rem 167
320 DATA009,255,240,007,255,224,001,255
:rem 166
330 DATA128,000,126,000,000,000,000,000
:rem 120
340 DATA000,126,000,001,255,128,007,255
:rem 153
350 DATA224,009,255,240,024,127,248,030
:rem 170
360 DATA031,248,063,135,252,063,227,252
:rem 177
370 DATA127,241,254,112,024,254,123,156
:rem 171
380 DATA254,127,062,126,062,126,124,063
:rem 176
390 DATA031,060,025,207,056,028,207,152
:rem 170
```



```

400 DATA014,031,144,007,255,224,001,255
      :rem 156
410 DATA128,000,126,000,000,000,000,000
      :rem 119
420 DATA000,126,000,001,255,128,007,207
      :rem 149
430 DATA224,015,031,144,030,127,152,028
      :rem 159
440 DATA255,056,057,159,060,057,158,124
      :rem 191
450 DATA120,014,126,127,060,254,127,056
      :rem 168
460 DATA254,127,241,254,063,227,252,063
      :rem 181
470 DATA135,252,030,031,248,024,127,248
      :rem 174
480 DATA009,255,240,007,255,224,001,255
      :rem 173
490 DATA128,000,126,000,000,000,000,000
      :rem 127
500 DATA000,000,000,000,000,126,000,001,255
      :rem 121
510 DATA128,007,255,224,009,255,240,024
      :rem 171
520 DATA127,248,030,031,248,063,135,252
      :rem 173
530 DATA063,227,252,127,241,254,120,024
      :rem 168
540 DATA254,121,156,254,121,254,126,060
      :rem 171
550 DATA062,124,063,159,060,025,207,056
      :rem 176
560 DATA028,207,152,014,031,144,007,255
      :rem 169
570 DATA224,001,255,128,000,126,000,000
      :rem 147
580 DATA000,000,000,000,126,000,001,255
      :rem 129
590 DATA128,007,255,224,009,255,240,024
      :rem 179
600 DATA127,248,030,031,248,063,135,252
      :rem 172
610 DATA063,227,252,127,241,254,126,024
      :rem 173
620 DATA254,124,204,254,121,254,126,056
      :rem 172
630 DATA062,124,057,159,060,025,207,056
      :rem 178
640 DATA028,207,152,014,031,144,007,255
      :rem 168
650 DATA224,001,255,128,000,126,000,000
      :rem 146
660 DATA000,126,000,001,255,128,007,255
      :rem 158
670 DATA224,012,007,144,025,207,152,031
      :rem 162
680 DATA159,056,063,063,060,062,126,124
      :rem 182
690 DATA126,126,126,124,252,254,124,248
      :rem 184
700 DATA254,127,241,254,063,227,252,063
      :rem 178
710 DATA135,252,030,031,248,024,127,248
      :rem 171
720 DATA009,255,240,007,255,224,001,255
      :rem 170
730 DATA128,000,126,000,000,000,000,000
      :rem 124
740 DATA000,126,000,001,255,128,007,015
      :rem 151
750 DATA224,014,103,144,028,231,152,028
      :rem 166

```

```

760 DATA207,056,062,031,060,060,206,124
      :rem 166
770 DATA121,206,126,121,156,254,124,056
      :rem 174
780 DATA254,127,241,254,063,227,252,063
      :rem 186
790 DATA135,252,030,031,248,024,127,248
      :rem 179
800 DATA009,255,240,007,255,224,001,255
      :rem 169
810 DATA128,000,126,000,000,000,000,000
      :rem 123

```

ABC del linguaggio macchina programma 5

```

10 FORI=49152TO49285:READB:POKEI,B:NEXT
      :rem 179
20 PRINT"[CLR]{WHT}":POKE53281,0:POKE5328
  0,0:V=53248:GOSUB210      :rem 122
30 PRINT"[HOME]{ 11 GIU' }[<L>][< 38 +>]
  [<J>]"      :rem 17
40 FORK=1TO2:PRINT"[SU][<L>][< 38 +>]
  [<J>]:NEXT      :rem 253
50 DIMX(16),Y(16):FORI=0TO15:X(I)=0:Y(I)=
  0      :rem 159
60 NEXT:FORI=0TO7:POKE832+I*2,INT((RND(I,
  *160)/10)*10+90      :rem 211
70 POKE848+I*2,INT((RND(I)*160)/10)*10+90
      :rem 177
80 POKE849+I*2,20      :rem 154
90 POKE833+I*2,200      :rem 196
100 POKE880+I,I+1:POKE888+I,15-I:rem 50
110 POKE864+I,248+I:POKE872+I,240+I:POKEV
  +21,255:NEXT:SYS49152      :rem 149
120 FORJ=0TO1:FORI=7TO0STEP-1:IFJ=1THEN14
  0      :rem 149
130 POKE833+I*2,INT((RND(I)*70)/10)*10+50
  :GOTO150      :rem 173
140 POKE849+I*2,INT((160+RND(I)*90)/10)*1
  0      :rem 224
150 GOSUB1650:FORK=1TO5:GOSUB1590:NEXTK,I
  ,J      :rem 204
160 FORP=1TO5:GOSUB1590:NEXT      :rem 22
170 FORJ=1TO0STEP-1:FORI=0TO7:IFJ=1THEN19
  0      :rem 159
180 POKE833+I*2,200:GOTO200      :rem 249
190 POKE849+I*2,20      :rem 204
200 GOSUB1710:FORK=1TO5:GOSUB1590:NEXTK,I
  ,J:GOTO120      :rem 203
210 FORK=0TO15:FORJ=0TO63:READB:POKE(255-
  K)*64+J,B:NEXT:RETURN      :rem 8
220 REM DATA ROUTINE      :rem 184
230 DATA120,173,20,3,141,132,192,173,21,3
  ,141,133,192,169,43,141,20,3,169,192
      :rem 230
240 DATA141,21,3,169,1,141,26,208,169,0,1
  41,18,208,173,17,208,41,127,141,17,20
  8      :rem 82
250 DATA88,96,173,25,208,41,1,240,81,141,
  25,208,173,18,208,208,35,169,145
      :rem 74
260 DATA141,18,208,162,15,189,64,3,157,0,
  208,138,24,106,168,185,96,3,153,248,7
      :rem 70
270 DATA185,112,3,153,39,208,202,16,231,7
  6,125,192,169,0,141,18,208,162,15
      :rem 105
280 DATA189,80,3,157,0,208,138,24,106,168
  ,185,104,3,153,248,7,185,120,3
      :rem 224

```



```

290 DATA153,39,208,202,16,231,104,168,104
    ,170,104,64,76,177,234      :rem 82
300 REM DATA SPRITES           :rem 187
310 DATA000,126,000,001,255,128,007,223
    :rem 145
320 DATA224,015,159,144,030,063,152,028
    :rem 167
330 DATA063,056,063,063,060,062,126,124
    :rem 168
340 DATA126,126,126,126,124,254,120,024
    :rem 164
350 DATA254,127,241,254,063,227,252,063
    :rem 179
360 DATA135,252,030,031,248,024,127,248
    :rem 172
370 DATA009,255,240,007,255,224,001,255
    :rem 171
380 DATA128,000,126,000,000,000,000,000
    :rem 125
390 DATA000,126,000,001,255,128,007,255
    :rem 158
400 DATA224,014,031,144,028,207,152,025
    :rem 158
410 DATA207,056,063,031,060,062,126,124
    :rem 162
420 DATA124,254,126,121,220,254,120,024
    :rem 155
430 DATA254,127,241,254,063,227,252,063
    :rem 178
440 DATA135,252,030,031,248,024,127,248
    :rem 171
450 DATA009,255,240,007,255,224,001,255
    :rem 170
460 DATA128,000,126,000,000,000,000,000
    :rem 124
470 DATA000,126,000,001,255,128,007,255
    :rem 157
480 DATA224,009,255,240,024,127,248,030
    :rem 174
490 DATA031,248,063,135,252,063,227,252
    :rem 181
500 DATA127,241,254,112,024,254,123,156
    :rem 166
510 DATA254,127,062,126,062,126,124,063
    :rem 171
520 DATA031,060,025,207,056,028,207,152
    :rem 165
530 DATA014,031,144,007,255,224,001,255
    :rem 160
540 DATA128,000,126,000,000,000,000,000
    :rem 123
550 DATA000,126,000,001,255,128,007,207
    :rem 153
560 DATA224,015,031,144,030,127,152,028
    :rem 163
570 DATA255,056,057,159,060,057,158,124
    :rem 195
580 DATA120,014,126,127,060,254,127,056
    :rem 172
590 DATA254,127,241,254,063,227,252,063
    :rem 185
600 DATA135,252,030,031,248,024,127,248
    :rem 169
610 DATA009,255,240,007,255,224,001,255
    :rem 168
620 DATA128,000,126,000,000,000,000,000
    :rem 122
630 DATA000,000,000,000,126,000,001,255
    :rem 125
640 DATA128,007,255,224,009,255,240,024
    :rem 175

```

```

650 DATA127,248,030,031,248,063,135,252
    :rem 177
660 DATA063,227,252,127,241,254,120,024
    :rem 172
670 DATA254,121,156,254,121,254,126,060
    :rem 175
680 DATA062,124,063,159,060,025,207,056
    :rem 180
690 DATA028,207,152,014,031,144,007,255
    :rem 173
700 DATA224,001,255,128,000,126,000,000
    :rem 142
710 DATA000,000,000,000,126,000,001,255
    :rem 124
720 DATA128,007,255,224,009,255,240,024
    :rem 174
730 DATA127,248,030,031,248,063,135,252
    :rem 176
740 DATA063,227,252,127,241,254,126,024
    :rem 177
750 DATA254,124,204,254,121,254,126,056
    :rem 176
760 DATA062,124,057,159,060,025,207,056
    :rem 182
770 DATA028,207,152,014,031,144,007,255
    :rem 172
780 DATA224,001,255,128,000,126,000,000
    :rem 150
790 DATA000,126,000,001,255,128,007,255
    :rem 162
800 DATA224,012,007,144,025,207,152,031
    :rem 157
810 DATA159,056,063,063,060,062,126,124
    :rem 177
820 DATA126,126,126,124,252,254,124,248
    :rem 179
830 DATA254,127,241,254,063,227,252,063
    :rem 182
840 DATA135,252,030,031,248,024,127,248
    :rem 175
850 DATA009,255,240,007,255,224,001,255
    :rem 174
860 DATA128,000,126,000,000,000,000,000
    :rem 128
870 DATA000,126,000,001,255,128,007,015
    :rem 155
880 DATA224,014,103,144,028,231,152,028
    :rem 170
890 DATA207,056,062,031,060,060,206,124
    :rem 170
900 DATA121,206,126,121,156,254,124,056
    :rem 169
910 DATA254,127,241,254,063,227,252,063
    :rem 181
920 DATA135,252,030,031,248,024,127,248
    :rem 174
930 DATA009,255,240,007,255,224,001,255
    :rem 173
940 DATA128,000,126,000,000,000,000,000
    :rem 127
950 DATA000,126,000,001,255,128,007,255
    :rem 160
960 DATA224,009,248,112,025,243,056,028
    :rem 181
970 DATA243,152,060,249,156,062,124,028
    :rem 185
980 DATA126,127,158,127,051,062,127,024
    :rem 184
990 DATA126,127,143,254,063,199,252,063
    :rem 196
1000 DATA225,252,031,248,120,031,254,024
    :rem 203

```

```

1010 DATA015,255,144,007,255,224,001,255
      :rem 211
1020 DATA128,000,126,000,000,000,000,000
      :rem 165
1030 DATA000,126,000,001,255,128,007,255
      :rem 198
1040 DATA224,015,255,208,029,135,200,025
      :rem 212
1050 DATA051,152,035,051,156,050,115,060
      :rem 206
1060 DATA102,115,062,102,102,126,102,100
      :rem 186
1070 DATA126,103,008,254,063,241,252,063
      :rem 216
1080 DATA195,252,031,015,248,028,063,248
      :rem 232
1090 DATA012,255,240,007,255,224,001,255
      :rem 213
1100 DATA128,000,126,000,000,000,000,000
      :rem 164
1110 DATA000,126,000,001,255,128,007,255
      :rem 197
1120 DATA224,015,255,208,030,247,200,028
      :rem 210
1130 DATA231,152,049,143,156,057,207,060
      :rem 221
1140 DATA115,159,062,115,158,126,115,156
      :rem 227
1150 DATA126,115,152,254,063,241,252,063
      :rem 218
1160 DATA195,252,031,015,248,028,063,248
      :rem 231
1170 DATA012,255,240,007,255,224,001,255
      :rem 212
1180 DATA128,000,126,000,000,000,000,000
      :rem 172
1190 DATA000,126,000,001,255,128,007,255
      :rem 205
1200 DATA224,015,255,208,029,135,200,025
      :rem 210
1210 DATA051,152,035,243,156,051,231,060
      :rem 207
1220 DATA103,143,062,103,062,126,102,124
      :rem 198
1230 DATA126,100,008,254,063,241,252,063
      :rem 211
1240 DATA195,252,031,015,248,028,063,248
      :rem 230
1250 DATA012,255,240,007,255,224,001,255
      :rem 211
1260 DATA128,000,126,000,000,000,000,000
      :rem 171
1270 DATA000,126,000,001,255,128,007,255
      :rem 204
1280 DATA224,015,255,208,029,131,200,025
      :rem 214
1290 DATA051,152,035,231,156,051,143,060
      :rem 214
1300 DATA103,231,062,103,230,126,100,204
      :rem 189
1310 DATA126,102,024,254,063,241,252,063
      :rem 210
1320 DATA195,252,031,015,248,028,063,248
      :rem 229
1330 DATA012,255,240,007,255,224,001,255
      :rem 210
1340 DATA128,000,126,000,000,000,000,000
      :rem 170
1350 DATA000,126,000,001,255,128,007,255
      :rem 203
1360 DATA224,015,255,208,029,207,200,025
      :rem 217

```

```

1370 DATA159,152,035,063,156,050,127,060
      :rem 226
1380 DATA102,079,062,100,006,126,103,204
      :rem 207
1390 DATA126,103,200,254,063,241,252,063
      :rem 215
1400 DATA195,252,031,015,248,028,063,248
      :rem 228
1410 DATA012,255,240,007,255,224,001,255
      :rem 209
1420 DATA128,000,126,000,000,000,000,000
      :rem 169
1430 DATA000,126,000,001,255,128,007,255
      :rem 202
1440 DATA224,015,255,208,029,003,200,025
      :rem 210
1450 DATA051,152,034,127,156,050,015,060
      :rem 212
1460 DATA103,231,062,100,230,126,100,204
      :rem 193
1470 DATA126,102,024,254,063,241,252,063
      :rem 217
1480 DATA195,252,031,015,248,028,063,248
      :rem 236
1490 DATA012,255,240,007,255,224,001,255
      :rem 217
1500 DATA128,000,126,000,000,000,000,000
      :rem 168
1510 DATA000,126,000,001,255,128,007,255
      :rem 201
1520 DATA224,015,255,208,029,135,200,025
      :rem 215
1530 DATA051,152,034,127,156,050,015,060
      :rem 211
1540 DATA100,231,062,100,230,126,100,204
      :rem 189
1550 DATA126,102,024,254,063,241,252,063
      :rem 216
1560 DATA195,252,031,015,248,028,063,248
      :rem 235
1570 DATA012,255,240,007,255,224,001,255
      :rem 216
1580 DATA128,000,126,000,000,000,000,000
      :rem 176
1590 FORW=0TO15:X=PEEK(832+W*2):Y=PEEK(83
      3+W*2) :rem 5
1600 IFX=250THENX(W)=-1:GOTO1620:rem 136
1610 IFX=90THENX(W)=+1 :rem 29
1620 IF(Y=120ANDW<8)OR(Y>=220ANDW>7) THENY
      (W)=-1:GOTO1640 :rem 114
1630 IF(Y=50ANDW<8)OR(Y=160ANDW>7) THENY(W)
      =+1 :rem 202
1640 X=X+X(W)*10:Y=Y+Y(W)*10:POKE(832+W*2
      ),X:POKE(833+W*2),Y:NEXT:RETURN :rem 85
1650 X=RND(TI):Y=RND(TI) :rem 213
1660 X(I+J*8)=-1:IFX>.33THENX(I+J*8)=0 :rem 137
1670 IFX>.66THENX(I+J*8)=+1 :rem 30
1680 Y(I+J*8)=-1:IFY>.33THENY(I+J*8)=0 :rem 142
1690 IFY>.66THENY(I+J*8)=+1 :rem 34
1700 RETURN :rem 168
1710 X(I+J*8)=0:Y(I+J*8)=0:RETURN :rem 138

```


Dalla grande edicola Jackson Tutto sul personal computer

PERSONAL O

In questo numero:

Un video per l'acciaio
Il problema del commesso viaggiatore
Programmare in "C"
La banca in casa
Assemblatori MsDos - Pcos

BIT

In questo numero:

Bitest:
Atari 130XE
Slalom parallelo:
HP PC Integrato
contro
Grid Compass II
Speciale:
Data-base
Corso di Assembly
per C64

PC MAGAZINE

In questo numero:

Sfogliando le Pagine Gialle
Execuision
dBase III
L'esperienza giapponese

PERSONAL SOFTWARE

In questo numero:

Easy-Video per C64
Geo-Race per Spectrum
Data-base per C16
Disegnatore per MSX

Personal-O/PC/Bit/Personal Software
sono pubblicazioni firmate:

GRUPPO EDITORIALE JACKSON

via Rosellini, 12-20124 Milano



Piccola guida per l'input dei programmi

Molti dei programmi listati da **SUPERCOMMODORE** contengono particolari caratteri di controllo (controllo cursore, tasti colore, video-inverse, ecc.). Per rendere chiaro ciò che dev'essere battuto quando è necessario inserire uno dei suddetti caratteri sono state stabilite le seguenti convenzioni.

Generalmente i listati per VIC 20 e C64 contengono delle «parole» racchiuse tra parentesi graffe {}; tali parole rappresentano particolari caratteri di controllo: {GIU} significa premere il simbolo del cursore verso il basso. {5 SPAZI} vuol dire battere 5 volte la barra-spazio.

Per indicare che un tasto dev'essere «shiftato», cioè premuto insieme al tasto SHIFT, nel listato il simbolo di questo tasto sarà sottolineato.

Per esempio S significa che dev'essere premuto il tasto S mentre è abbassato il tasto SHIFT.

La manovra farà apparire sullo schermo un piccolo «cuore».

Trovando un simbolo sottolineato chiuso tra parentesi graffe (esempio {10 N}) dovrà essere interpretato come «premere il carattere indicato per il numero di volte che lo precede nelle parentesi»: nel nostro esempio premere 10 volte il tasto N «shiftato».

Se il tasto da premere è indicato tra le parentesi [()], vuol dire che lo stesso dovrà essere premuto mentre è premuto il tasto «Commodore» (il tasto «Commodore» è quello posto nell'angolo in basso a sinistra).

Se il tasto scritto tra [()] è preceduto da un numero, ciò significa che il tasto dev'essere premuto per il numero di volte indicato.

Raramente si potrà trovare un carattere alfabetico solitario racchiuso tra parentesi graffe. Tale carattere sul C64 può essere battuto mentre è premuto il tasto CTRL.

Ad esempio {A} sta ad indicare la sequenza CTRL-A. A proposito del «modo virgolette» è nota la possibilità di muovere il cursore sullo schermo con i tasti CRSR. Ogni tanto i programmatori desiderano muovere il cursore durante l'esecuzione del programma. È per questo motivo che nei programmi si troveranno dei {SIN}, {HOME} e {BLU}.

L'unico modo perché il calcolatore distingua il comando diretto da quello inserito nel programma è il «modo virgolette». Premendo il tasto «virgolette» (tasti SHIFT 2) il calcolatore si predispose appunto in «modo virgolette».

Battendo un carattere qualsiasi, cercando poi di correggerlo muovendo il cursore, si otterrà solo un tratto verticale in un quadratino inverso. Questo è infatti il simbolo del cursore a sinistra nel «modo virgolette». L'unico comando di editing non utilizzabile all'interno di un programma è il DEL. Battendo nuovamente le «virgolette» il calcolatore lascia il «modo virgolette». Si può accedere al «modo virgolette» quando inserite spazi nella linea.

In ogni caso, il sistema più semplice per uscire dal «modo virgolette» è quello di premere il tasto RETURN.

Utilizzare la tabella che segue quando è necessario inserire comandi relativi al cursore e ai tasti colore.

Quando leggete:	Premete:	Vedrete:	Quando leggete:	Premete:	Vedrete:	Quando leggete:	Premete:	Vedrete:
{CLR}	SHIFT CLR/HOME		{CYN}	CTRL 4		[<7>]	CTRL 7	
{HOME}	CLR/HOME		{PUR}	CTRL 5		[<8>]	CTRL 8	
{SU}	SHIFT ↑ CRSR		{GRN}	CTRL 6		{F1}	F1	
{GIU}	↓ CRSR		{BLU}	CTRL 7		{F2}	F2	
{SIN}	SHIFT ← CRSR		{YEL}	CTRL 8		{F3}	F3	
{DES}	→ CRSR		[<1>]	CTRL 1		{F4}	F4	
{RVS}	CTRL 9		[<2>]	CTRL 2		{F5}	F5	
{OFF}	CTRL 8		[<3>]	CTRL 3		{F6}	F6	
{BLK}	CTRL 1		[<4>]	CTRL 4		{F7}	F7	
{WHT}	CTRL 2		[<5>]	CTRL 5		{F8}	F8	
{RED}	CTRL 3		[<6>]	CTRL 6				

MLX per VIC 20 e C64 (versione 2.0 per C64)

di C. Brannon
trad. e adatt.
di M. Cristofalo Grizzi
e F. Stella

M LX è un programma che permette di inserire listati in linguaggio macchina esenti in modo assoluto da errori e senza la perdita di tempo del dover battere e controllare lunghe sequenze di istruzioni DATA.

Molti dei nostri listati di programmi in linguaggio macchina hanno il formato MLX (li riconoscete dal fatto che sono esclusivamente numerici) e richiedono quindi che MLX sia caricato in memoria ed eseguito prima della battitura del listato.

MLX vi chiederà l'indirizzo della locazione di partenza e quello della locazione finale del programma da caricare. Questi valori sono sempre indicati nell'articolo che accompagna il listato. MLX vi segnala automaticamente gli errori di battitura MENTRE STATE DIGITANDO IL LISTATO e vi chiede di reinserire la linea errata. L'ultimo numero battuto di ogni

linea rappresenta il checksum e viene visualizzato in reverse.

Sono inoltre disponibili altri comandi, quali: SHIFT-N nuovo indirizzo: permette di cambiare l'indirizzo della linea che volete battere ed è utile nel caso si inseriscano i listati in più parti.

SHIFT-D display: lista i dati inseriti tra due indirizzi di inizio e fine.

SHIFT-L load: carica su file da nastro o SHIFT-S save: salva su nastro o disco un file in formato MLX.

Una volta battuto il listato, e salvatolo tramite MLX, si potrà caricare direttamente il programma con un'istruzione LOAD "nome del file",1,1 per il registratore, oppure LOAD "nome del file",8,1 per l'unità a dischi, salvo diverse indicazioni date negli articoli corrispondenti ai vari programmi. Il programma sarà generalmente mandato in esecuzione con una SYS (indirizzo di partenza).

La versione originaria di MLX per C64 ha subito, dalla prima pubblicazione, diversi ritocchi e miglioramenti, fino alla versione 2.0 qui presentata.

È stato in primo luogo modificato il colore di fondo dello schermo e del bordo, in modo da risultare meno stancante alla vista; sono state ulteriormente perfezionate le routine in linguaggio macchina di salvataggio e caricamento dei programmi; è cambiata la forma del cursore e, cosa più importante, è stato aggiunto un tastierino numerico per gli utenti che si trovano più a loro agio con una diversa disposizione dei tasti.

Oltre a poter usare i soliti tasti numerici, sono stati ridefiniti alcuni tasti alfabetici, in modo da generare ugualmente dei numeri. I tasti ridefiniti sono i seguenti:

U I O	7 8 9
H J K L	diventano 0 4 5 6
M, .	1 2 3

Le persone abituate ad usare tali tastierini accresceranno senz'altro, in brevissimo tempo, la velocità di battitura dei programmi in linguaggio macchina.

MLX versione per VIC 20

```

100 PRINT" {CLR} {PUR}";CHR$(142);CHR$(8);
      :rem 181
101 POKE788,194:REM DISABILITA RUN/STOP
      :rem 144
110 PRINT" {RVS} { 14 SPAZI}"
      :rem 117
120 PRINT" {RVS} {DES} {OFF} [<*>] {RVS}
      {DES} {DES} { 2 SPAZI} [<*>] {OFF} [<*>]
      {RVS} {RVS} "
      :rem 191
130 PRINT" {RVS} {DES} [<G>] {DES}
      { 2 DES} {OFF} {RVS} [<*>] {OFF} [<*>]
      {RVS} {OFF}"
      :rem 122
140 PRINT" {RVS} { 14 SPAZI}"
      :rem 120
200 PRINT" { 2 GIU' } {PUR} {BLK} UN PROGRAMMA
      PER L'IN-TRODUZIONE ( 2 SPAZI) DI ROUT
      INEIN LINGUAGGIO MACCHINA";
      :rem 202
205 PRINT" A PROVA DI ERRORE ( 5 GIU' )"
      :rem 226
210 PRINT" {BLK} { 4 SU} INDIRIZZO DI PARTEN
      ZA":INPUTS:F=1-F:C$=CHR$(31+119*F)
      :rem 52
220 IFS<256ORS>32767THENGOSUB3000:GOTO210
      :rem 2
225 PRINT:PRINT:PRINT:PRINT
      :rem 123
230 PRINT" {BLK} { 3 SU} INDIRIZZO CONCLUSIV
      O":INPUTS:F=1-F:C$=CHR$(31+119*F)
      :rem 170
240 IFE<256ORE>32767THENGOSUB3000:GOTO230
      :rem 234
250 IFE<STHENPRINTC$;" {RVS} INDIRIZZO CONC
      LUSIVO <INDIRIZZO DI PARTENZA
      { 3 GIU' }":GOSUB1000:GOTO230:rem 27
260 PRINT:PRINT:PRINT
      :rem 179
300 PRINT" {CLR}";CHR$(14):AD=S
      :rem 56
310 PRINTRIGHT$( "0000"+MID$(STR$(AD),2),5
      );":":FORJ=1TO6
      :rem 234
320 GOSUB570:IFN=-1THENJ=J+N:GOTO320
      :rem 228
390 IFN=-211THEN710
      :rem 62
400 IFN=-204THEN790
      :rem 64
410 IFN=-206THENPRINT:PRINT" {GIU' } INSERIS
      CI UN NUOVO { 4 SPAZI} INDIRIZZO";INPU
      TZZ
      :rem 42

```

```

415 IFN=-206THENIFZZ<SORZZ>ETHENPRINT"
      {RVS}ESCE DAL CAMPO DI VA-" :rem 150
416 IFN=-206THENIFZZ<SORZZ>ETHENPRINT"
      {RVS}LORI INDICATO":GOSUB1000:GOTO410
      :rem 105
417 IFN=-206THENAD=ZZ:PRINT:GOTO310
      :rem 238
420 IFN<>-196THEN480
      :rem 133
430 PRINT:INPUT"LISTATO:DA";F:PRINT,"
      { 2 SIN}A";:INPUTT
      :rem 29
440 IFF<SORF>EORT<SORT>ETHENPRINT"MINIMO"
      ;S;" {SIN}, MASSIMO";E:GOTO430
      :rem 36
450 FORI=FTOTSTEP6:PRINT:PRINTRIGHT$( "000
      0"+MID$(STR$(I),2),5);":":
      :rem 30
455 FORK=OTO5:N=PEEK(I+K):IFK=3THENPRINTS
      PC(10);
      :rem 34
457 PRINTRIGHT$( "00"+MID$(STR$(N),2),3);"
      ,";
      :rem 157
460 GETA$:IFA$>" THENPRINT:PRINT:GOTO310
      :rem 25
470 NEXTK:PRINTCHR$(20);:NEXTI:PRINT:PRIN
      T:GOTO310
      :rem 50
480 IFN<0THENPRINT:GOTO310
      :rem 168
490 A(J)=N:NEXTJ
      :rem 199
500 CKSUM=AD-INT(AD/256)*256:FORI=1TO6:CK
      SUM=(CKSUM+A(I))AND255:NEXT
      :rem 200
510 PRINTCHR$(18);:GOSUB570:PRINTCHR$(20)
      :rem 234
515 IFN=CKSUMTHEN530
      :rem 255
520 PRINT:PRINT"LA RIGA E' STATA INSE-RIT
      A IN MANIERA ERRATA"
      :rem 39
525 PRINT"RI-INSEIRILA":PRINT:GOSUB1000:G
      OTO310
      :rem 167
530 GOSUB2000
      :rem 218
540 FORI=1TO6:POKEAD+I-1,A(I):NEXT
      :rem 80
550 AD=AD+6:IFAD<ETHEN310
      :rem 212
560 GOTO710
      :rem 108
570 N=0:Z=0
      :rem 88

```



```

580 PRINT "[<+>]"; :rem 79
581 GETA$:IFA$="" THEN 581 :rem 95
585 PRINT CHR$(20);A=ASC(A$):IFA=13ORA=44 :rem 229
ORA=32THEN 670 :rem 229
590 IFA>128 THEN N=-A:RETURN :rem 137
600 IFA<>20 THEN 630 :rem 10
610 GOSUB 690:IFI=1AND T=44 THEN N=-1:PRINT " :rem 172
{SIN} {SIN}";:GOTO 690 :rem 109
620 GOTO 570 :rem 105
630 IFA<48ORA>57 THEN 580 :rem 106
640 PRINTA$:N=N*10+A-48 :rem 229
650 IFN>255 THEN A=20:GOSUB 1000:GOTO 600 :rem 71
:rem 114
660 Z=Z+1:IFZ<3 THEN 580 :rem 240
670 IFZ=0 THEN GOSUB 1000:GOTO 570 :rem 149
680 PRINT";":RETURN :rem 68
690 S%=PEEK(209)+256*PEEK(210)+PEEK(211) :rem 205
:rem 7
692 FORI=1 TO 3:T=PEEK(S%-I) :rem 30
695 IFT<>44AND T<>58 THEN POKES%-I,32:NEXT :rem 158
:rem 3
700 PRINT LEFT$( "{ 3 SIN}",I-1);:RETURN :rem 12
710 PRINT "{CLR}{RVS}*** SAVE *** :rem 124
{ 3 GIU' }" :rem 124
720 INPUT "{GIU'} NOME DEL FILE";F$ :rem 111
:rem 130
730 PRINT:PRINT "{ 2 GIU'}{RVS}N{OFF}ASTRO :rem 128
O {RVS}D{OFF}ISCO: (N/D)" :rem 102
740 GETA$:IFA$<>"N"ANDA$<>"D" THEN 740 :rem 231
:rem 206
750 DV=1-7*(A$="D"):IFDV=8 THEN F$="0":+F$ :rem 117
:rem 74
760 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$ :rem 130
):POKE 782,ZK/256 :rem 22
762 POKE 781,ZK-PEEK(782)*256:POKE 780,LEN( :rem 119
T$):SYS65469 :rem 69
763 POKE 780,1:POKE 781,DV:POKE 782,1:SYS654 :rem 12
66 :rem 124
765 POKE 254,S/256:POKE 253,S-PEEK(254)*256 :rem 119
:POKE 780,253 :rem 12
766 POKE 782,E/256:POKE 781,E-PEEK(782)*256 :rem 119
:SYS65496 :rem 124
770 IF (PEEK(783)AND 1) OR (ST AND 191) THEN 780 :rem 111
:rem 111

```

```

775 PRINT "{GIU'} OPERAZIONE CONCLUSA.":END :rem 152
:rem 8
780 PRINT "{GIU'} ERRORE DI REGISTRAZIONE- :rem 103
RIPROVA.":IFDV=1 THEN 720 :rem 115
781 OPEN 15,8,15:INPUT #15,E1$,E2$:PRINT E1$ :rem 212
;E2$:CLOSE 15:GOTO 720 :rem 215
782 GOTO 720 :rem 127
790 PRINT "{CLR}{RVS}*** LOAD *** :rem 2
{ 2 GIU' }" :rem 107
800 INPUT "{ 2 GIU' } NOME DEL FILE";F$ :rem 70
:rem 11
810 PRINT:PRINT "{ 2 GIU'}{RVS}N{OFF}ASTRO :rem 142
O {RVS}D{OFF}ISCO: (N/D)" :rem 43
820 GETA$:IFA$<>"N"ANDA$<>"D" THEN 820 :rem 102
:rem 231
830 DV=1-7*(A$="D"):IFDV=8 THEN F$="0":+F$ :rem 206
:rem 117
840 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$ :rem 74
):POKE 782,ZK/256 :rem 130
841 POKE 781,ZK-PEEK(782)*256:POKE 780,LEN( :rem 22
T$):SYS65469 :rem 119
845 POKE 780,1:POKE 781,DV:POKE 782,1:SYS654 :rem 12
66 :rem 124
850 POKE 780,0:SYS65493 :rem 111
860 IF (PEEK(783)AND 1) OR (ST AND 191) THEN 870 :rem 130
:rem 22
865 PRINT "{GIU'} OPERAZIONE CONCLUSA.":GOT :rem 119
O 310 :rem 12
870 PRINT "{GIU'} ERRORE DI CARICAMENTO. RIP :rem 117
ROVA. {GIU' }":IFDV=1 THEN 800 :rem 74
880 OPEN 15,8,15:INPUT #15,E1$,E2$:PRINT E1$ :rem 130
;E2$:CLOSE 15:GOTO 800 :rem 206
1000 REM CICALINO :rem 117
1001 POKE 36878,15:POKE 36874,190 :rem 74
1002 FORW=1 TO 300:NEXTW :rem 130
1003 POKE 36878,0:POKE 36874,0:RETURN :rem 22
:rem 119
2000 REM CAMPANELLO :rem 119
2001 FORW=15 TO 0 STEP -1:POKE 36878,W:POKE 368 :rem 12
76,240:NEXTW :rem 119
2002 POKE 36876,0:RETURN :rem 130
3000 PRINTC$;"{RVS}NON IN PAGINA ZERO O :rem 105
{ 2 SPAZI }SU ROM{ 4 GIU' }":GOTO 1000 :rem 23

```

MLX versione 2.0 per C64

```

100 PRINT "{CLR}{CYN}";CHR$(142);CHR$(8);: :rem 71
POKE 53280,0:POKE 53281,0 :rem 89
101 POKE 788,52:REM DISABILITA RUN/STOP :rem 176
:rem 176
110 PRINT "{RVS}{ 40 SPAZI}"; :rem 176
120 PRINT "{RVS}{ 15 SPAZI}{DES}{OFF}{<*>} :rem 250
~{RVS}{DES}{DES}{ 2 SPAZI}{<*>}{OFF} :rem 220
[<*>]~{RVS}~{RVS}{ 13 SPAZI}"; :rem 120
:rem 120
130 PRINT "{RVS}{ 15 SPAZI}{DES}{<N>}{<H>} :rem 235
{DES}{ 2 DES}{OFF}~{RVS}~{<*>}{OFF} :rem 180
[<*>]~{RVS}{ 13 SPAZI}"; :rem 180
140 PRINT "{RVS}{ 40 SPAZI}"; :rem 180
200 PRINT "{ 2 GIU'}{PUR}EDITOR DI CODICE :rem 126
MACCHINA VERSIONE 2.0{ 3 GIU' }" :rem 62
210 PRINT "[<+>]{ 2 SU} INDIRIZZO DI PARTE :rem 64
NZA{ 2 SPAZI}";:INPUTS:F=1-F:C$=CHR$( :rem 13
31+119*F) :rem 13
220 IFS<256OR(S>40960ANDS<49152)ORS>53247 :rem 235
THEN GOSUB 3000:GOTO 210 :rem 180
225 PRINT:PRINT:PRINT :rem 180

```

```

230 PRINT "[<+>]{ 2 SU} INDIRIZZO CONCLUSI :rem 92
VO{ 3 SPAZI}";:INPUTE:F=1-F:C$=CHR$(3 :rem 183
1+119*F) :rem 183
240 IFE<256OR(E>40960ANDE<49152)ORE>53247 :rem 200
THEN GOSUB 3000:GOTO 230 :rem 119
250 IFE<STHEN PRINTC$;"{RVS}INDIRIZZO CONC :rem 179
LUSIVO<INDIRIZZO INIZIALE" :rem 105
255 IFE<STHEN GOSUB 1000:GOTO 230 :rem 225
260 PRINT:PRINT:PRINT :rem 227
300 PRINT "{CLR}";CHR$(14):AD=S:POKEV+21,0 :rem 33
:rem 227
310 A=1:PRINT RIGHT$( "0000"+MID$(STR$(AD), :rem 33
2),5);":":FORJ=ATO6 :rem 33
315 FORJ=ATO6 :rem 33
320 GOSUB 570:IFN=-1 THEN J=J+N:GOTO 320 :rem 228
:rem 62
390 IFN=-211 THEN 710 :rem 64
400 IFN=-204 THEN 790 :rem 13
410 IFN=-206 THEN PRINT:INPUT "{GIU'} INSERI :rem 105
RE IL NUOVO INDIRIZZO";ZZ :rem 105
414 IFN=-206 THEN IFZZ<SORZZ>ETHEN PRINT " :rem 105
{RVS}ESCE DAL CAMPO DI VALORI INDICAT :rem 105
O" :rem 105

```



```

415 IFN=-206THENIFZZ<SORZZ>ETHENGOSUB1000
:GOTO410 :rem 202
417 IFN=-206THENAD=ZZ:PRINT:GOTO310 :rem 238
420 IFN<>-196THEN480 :rem 133
430 PRINT:INPUT"LISTATO:DA";F:PRINT"
{ 9 SPAZI}A";:INPUTT :rem 183
440 IFF<SORF>EORT<SORT>ETHENPRINT"MINIMO"
;S;" MASSIMO";E;"! [<5>":GOTO430 :rem 147
450 FORI=FTOTSTEP6:PRINT:PRINTRIGHTS("000
0"+MID$(STR$(I),2),5);": :rem 30
451 FOK=0TO5:N=PEEK(I+K):PRINTRIGHTS("00
"+MID$(STR$(N),2),3);": :rem 66
460 GETAS:IFAS>" THENPRINT:PRINT:GOTO310
:rem 25
470 NEXTK:PRINTCHR$(20);:NEXTI:PRINT:PRIN
T:GOTO310 :rem 50
480 IFN<0THENPRINT:GOTO310 :rem 168
490 A(J)=N:NEXTJ :rem 199
500 CKSUM=AD-INT(AD/256)*256:FORI=1TO6:CK
SUM=(CKSUM+A(I))AND255:NEXT :rem 200
510 PRINTCHR$(18);:GOSUB570:PRINTCHR$(146
); :rem 94
511 IFN=-1THENA=6:GOTO315 :rem 254
515 PRINTCHR$(20):IFN=CKSUMTHEN530 :rem 122
520 PRINT:PRINT"{RED}LA LINEA E' STATA IN
SERITA IN MANIERA" :rem 157
525 PRINT"ERRATA. RIPETERE [<5>":PRINT:GO
SUB1000:GOTO310 :rem 27
530 GOSUB2000 :rem 218
540 FORI=1TO6:POKEAD+I-1,A(I):NEXT:POKE54
272,0:POKE54273,0 :rem 227
550 AD=AD+6:IFAD<ETHEN310 :rem 212
560 GOTO710 :rem 108
570 N=0:Z=0 :rem 88
580 PRINT"< >"; :rem 81
581 GETAS:IFAS=" THEN581 :rem 95
582 AV=- (AS="M")-2*(AS="")-3*(AS=".")-4*
(AS="J")-5*(AS="K")-6*(AS="L") :rem 41
583 AV=AV-7*(AS="U")-8*(AS="I")-9*(AS="O"
):IFAS="H"THENA="0" :rem 134
584 IFAV>0THENA=CHR$(48+AV) :rem 134
585 PRINTCHR$(20);:A=ASC(AS):IFA=13ORA=44
ORA=32THEN670 :rem 229
590 IFA>128THENN=-A:RETURN :rem 137
600 IFA<>20THEN630 :rem 10
610 GOSUB690:IFI=1ANDT=44THENN=-1:PRINT"
{SIN} {SIN}";:GOTO690 :rem 172
620 GOTO570 :rem 109
630 IFA<48ORA>57THEN580 :rem 105
640 PRINTAS;:N=N*10+A-48 :rem 106
650 IFN>255THENA=20:GOSUB1000:GOTO600
:rem 229
660 Z=Z+1:IFZ<3THEN580 :rem 71
670 IFZ=0THENGOSUB1000:GOTO570 :rem 114
680 PRINT",";:RETURN :rem 240
690 S%=PEEK(209)+256*PEEK(210)+PEEK(211)
:rem 149
691 FORI=1TO3:T=PEEK(S%-I) :rem 67
695 IFT<>44ANDT<>58THENPOKES%-I,32:NEXT
:rem 205
700 PRINTLEFTS("{ 3 SIN}",I-1);:RETURN
:rem 7
710 PRINT"{CLR} {RVS} *** SAVE ***
{ 3 GIU' }" :rem 236
715 PRINT"{ 2 GIU' }PREMERE {RVS}RETURN
{OFF} PER USCIRE DAL SAVE{GIU' }"
:rem 103

```

```

720 FS="":INPUT"{GIU'}NOME DEL FILE";FS:I
FFS="":THENPRINT:PRINT:GOTO310 :rem 42
730 PRINT:PRINT"{ 2 GIU' } {RVS}N{OFF}ASTRO
O {RVS}D{OFF}ISCO: (N/D)" :rem 128
740 GETAS:IFAS<>"N"ANDAS<>"D"THEN740
:rem 30
750 DV=1-7*(AS="D"):IFDV=8THENFS="0":+FS:
OPEN15,8,15,"S"+FS:CLOSE15 :rem 212
760 TS=FS:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
):POKE782,ZK/256 :rem 3
762 POKE781,ZK-PEEK(782)*256:POKE780,LEN(
T$):SYS65469 :rem 109
763 POKE780,1:POKE781,DV:POKE782,1:SYS654
66 :rem 69
765 K=S:POKE254,K/256:POKE253,K-PEEK(254)
*256:POKE780,253 :rem 17
766 K=E+1:POKE782,K/256:POKE781,K-PEEK(78
2)*256:SYS65496 :rem 235
770 IF(PEEK(783)AND1)OR(191ANDST)THEN780
:rem 111
775 PRINT"{GIU'}FATTO. {GIU'}":GOTO310
:rem 201
780 PRINT"{GIU'}ERRORE NEL SAVE-RIPROVA!
:IFDV=1THEN720 :rem 104
781 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
;E2$:CLOSE15:GOTO720 :rem 103
790 PRINT"{CLR} {RVS} *** LOAD ***
{ 2 GIU' }" :rem 212
795 PRINT"{ 2 GIU' }PREMERE {RVS}RETURN
{OFF} PER USCIRE DAL LOAD{GIU' }"
:rem 96
800 FS="":INPUT"{ 2 GIU' }NOME DEL FILE";F
$:IFFS="":THENPRINT:GOTO310 :rem 115
810 PRINT:PRINT"{ 2 GIU' } {RVS}N{OFF}ASTRO
O {RVS}D{OFF}ISCO: (N/D)" :rem 127
820 GETAS:IFAS<>"N"ANDAS<>"D"THEN820
:rem 28
830 DV=1-7*(AS="D"):IFDV=8THENFS="0":+FS
:rem 157
840 TS=FS:ZK=PEEK(53)+256*PEEK(54)-LEN(T$
):POKE782,ZK/256 :rem 2
841 POKE781,ZK-PEEK(782)*256:POKE780,LEN(
T$):SYS65469 :rem 107
845 POKE780,1:POKE781,DV:POKE782,1:SYS654
66 :rem 70
850 POKE780,0:SYS65493 :rem 11
860 IF(PEEK(783)AND1)OR(191ANDST)THEN870
:rem 111
865 PRINT"{GIU'}FATTO.":GOTO310 :rem 184
870 PRINT"{GIU'}ERRORE NEL LOAD-RIPETI!
{GIU'}":IFDV=1THEN800 :rem 19
880 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$
;E2$:CLOSE15:GOTO800 :rem 102
1000 REM CICALINO :rem 231
1001 POKE54296,15:POKE54277,45:POKE54278,
165 :rem 207
1002 POKE54276,33:POKE54273,6:POKE54272,5
:rem 42
1003 FORT=1TO200:NEXT:POKE54276,32:POKE54
273,0:POKE54272,0:RETURN :rem 202
2000 REM CAMPANELLO :rem 130
2001 POKE54296,15:POKE54277,0:POKE54278,2
47 :rem 152
2002 POKE54276,17:POKE54273,40:POKE54272,
0 :rem 86
2003 FORT=1TO100:NEXT:POKE54276,16:RETURN
:rem 57
3000 PRINTCS;"{RVS} NON IN PAGINA ZERO O
SU{DES}ROM ":GOTO1000 :rem 240

```


REM:HW

Hardware

Scopriamo insieme la stampante MPS-802

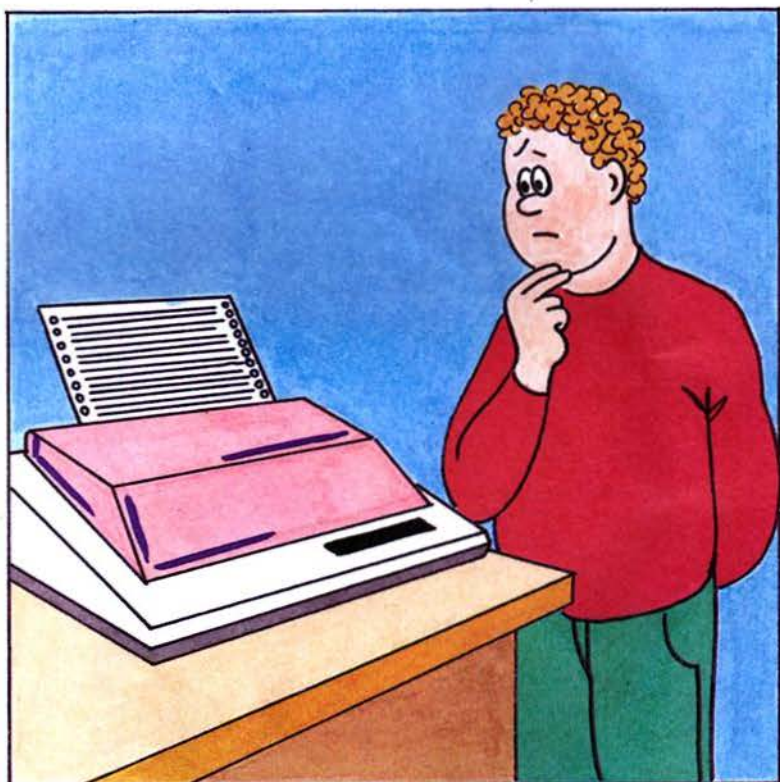
di E. Sarcina

La MPS-802 è una buona stampante, che però a prima vista sembrerebbe nata solo per riprodurre testi e tabelle. Andiamo insieme alla scoperta delle sue doti nascoste: vi accorgete che con questa periferica è possibile anche fare dell'ottima grafica. Alla fine di questo articolo troverete due listati: il primo vi consente di "programmare" la MPS-802 secondo le vostre esigenze, ed il secondo è una routine in linguaggio macchina che stampa la copia su carta di una schermata ad alta risoluzione del C64.

Molti lettori che posseggono una stampante Commodore MPS-802 ci scrivono lamentando la sua incompatibilità con le altre stampanti Commodore per quanto riguarda la creazione di caratteri personalizzati o, più in generale, chiedendoci se, e in quale modo, sia possibile gestire con questa periferica la grafica "punto a punto", cioè la stampa in alta risoluzione.

Pensiamo quindi di andare incontro alle esigenze di molti utenti Commodore affrontando tale argomento: vedremo insieme come sfruttare al meglio e nel modo più semplice le caratteristiche di questa stampante, colmando magari alcune lacune del manuale fornito dalla casa produttrice, che liquida in sole due pagine le possibilità grafiche di questa macchina.

Consideriamo brevemente le caratteristiche della MPS-802. Si tratta di una stampante bidirezionale, che può usare sia tabulati a modulo continuo che fogli singoli; ha buone caratteristiche di velocità (col suoi 60 caratteri al secondo riempie un foglio formato A4 in circa un minuto); è una periferica "intelligente", cioè governata da un microprocessore, per cui possiede una memoria interna e, soprattutto, un sistema operati-



vo, vale a dire un programma residente in ROM che gestisce il funzionamento della macchina in tutti i suoi aspetti. Proprio la complessità di questo sistema operativa costituisce la caratteristica che, a nostro avviso, eleva la MPS-802 un gradino al di sopra delle altre stampanti economiche disponibili per i computer Commodore. Tale programma non serve solo a gestire lo spostamento della testina di stampa, ma bensì è in grado di accettare una vasta gamma di comandi e di informazioni provenienti

dal computer, che l'utente invia alla stampante per mezzo di file con determinati indirizzi secondari.

Gli indirizzi secondari

Come saprete, la sintassi del comando OPEN del CBM BASIC è

OPEN nfi,np,ls

dove "nfi" è il numero del file logico (qualsiasi numero compreso tra 1 e 255), "np" è il numero della periferica,

detto anche indirizzo primario (nel nostro caso 4) e "is" è l'indirizzo secondario.

Nel caso della MPS-802 l'importanza dell'indirizzo secondario è fondamentale: esso serve a comunicare alla stampante come deve interpretare le informazioni provenienti dal computer. Sono possibili ben dieci indirizzi secondari:

- 0 - Stampa i dati esattamente come ricevuti, usando il set maiuscolo/grafico. Si tratta del valore default, e può essere omesso. Utilizzato anche dalle altre stampanti Commodore.

- 1 - Stampa i dati secondo il formato definito precedentemente.

- 2 - Definisce il formato dei dati da stampare.

- 3 - Regola il numero di righe stampate per pagina.

- 4 - Abilita la stampa dei messaggi di errore.

- 5 - Definisce un carattere programmabile.

- 6 - Regola la spaziatura tra le righe.

- 7 - Stampa i dati esattamente come ricevuti, usando il set minuscolo/maiuscolo. Utilizzato anche dalle altre stampanti Commodore.

- 9 - Disabilita la stampa dei messaggi di errore.

- 10 - Reinizializza la stampante, riportandola alle condizioni di accensione.

Per comodità possiamo definire "file di stampa" quelli con indirizzi secondari 0-1-7 e "file di controllo" tutti gli altri. È appunto la gestione delle informazioni tramite gli indirizzi secondari ciò che distingue la MPS-802 dalle altre stampanti Commodore, limitando la sua compatibilità con il software ad esse dedicato. Bisogna dire che tale incompatibilità si avverte solo con programmi che facciano uso di caratteri ridefiniti o effettuino stampa ad alta risoluzione; in tutti gli altri casi non si riscontra infatti alcun inconveniente.

Il motivo principale di questa mancanza di compatibilità, tuttavia, risiede a monte della questione degli indirizzi secondari: sta nel fatto che la matrice-carattere della MPS-802 è formata da 8x8 punti, e non 6x7 come nelle altre stampanti Commodore. Ciò rende ovviamente diverso il lavoro di codifica di un carattere programmabile. Inoltre la MPS-802 consente di definire (indirizzo secondario 5) un carattere personalizzato alla volta, ed ogni riga stampata ne può contenere uno solo. Queste sembrerebbero limitazioni molto pesanti, ma, come vedremo più avanti, esiste il modo di aggirarle.

Consideriamo ora i pregi di questa stampante: le informazioni inviabili tramite gli indirizzi secondari consentono di "maneggiare" i dati con molta elasticità, in quanto è possibile programmare la tabulazione della stampa (eventualmente con separazione di pagina) e regolare con precisione la distanza tra le righe, adattandola a qualunque tipo di foglio o modulo: tutte cose che con le altre stampanti economiche Commodore

non sono possibili, o perlomeno presuppongono la presenza di software apposito nella memoria del computer, diminuendo quindi la RAM disponibile per i programmi dell'utente. Inoltre la qualità della stampa, grazie al maggior numero di punti (64 contro 42) che compongono il carattere, è indubbiamente migliore rispetto alle concorrenti; infine i punti sono "quadrati" e a contatto l'uno con l'altro, dando maggior compattezza alla scrittura, specialmente rispetto alla cugina MPS-803.

Programmiamo la MPS-802

La MPS-802 è quindi una stampante "programmabile". È infatti possibile formattare la stampa di parole e numeri per creare tabelle (indirizzi secondari 1 e 2), opzione molto interessante, ma che tuttavia esula dall'argomento di questo articolo; si può inoltre selezionare la distanza tra le righe ed il numero di righe per pagina. Vorremmo spendere due parole proprio riguardo a queste ultime possibilità.

Per determinare la distanza tra due righe è sufficiente inviare alla stampante, sia da programma che in modo diretto, i seguenti comandi:

OPEN6,4,6:PRINT #6,CHR\$(d):CLOSE6

dove "d" è una variabile o un valore numerico compreso tra 0 e 127.

Questo valore rappresenta il numero di "passi" di cui la stampante fa avanzare la carta ogni volta che riceve, tramite un file di stampa e cioè con indirizzi secondari 0-1-7, il carattere CHR\$(13) ("carriage return" = vai a capo). Ricordiamo che un CHR\$(13) viene inviato automaticamente alla stampante (o anche allo schermo) ogni volta che una istruzione PRINT o PRINT # non termina con una virgola o con un punto e virgola.

Cerchiamo di capire cosa sono questi passi e, soprattutto, di misurare quanto sono lunghi. Un passo costituisce una piccolissima frazione di pollice (il pollice è un'unità di misura anglosassone, che equivale a 2.54 centimetri); sulla lunghezza dei passi, tuttavia, ci troviamo in disaccordo col manuale fornito insieme alla MPS-802. Esso afferma che un pollice è diviso in 144 passi, mentre noi ne abbiamo misurati, usando la nostra stampante, 216. I nostri passi sono quindi lunghi 1/216 di pollice (circa 0.0046 pollici), per cui risultano esattamente una volta e mezzo più corti di quelli del manuale. Non sappiamo se questa discrepanza sia dovuta a due diverse "tarature" delle stampanti in commercio o semplicemente ad un errore del manuale: ci limitiamo a segnalarla.

In base alle prove effettuate sulla nostra MPS-802 abbiamo costruito una tabella con i valori da assegnare a "d" per ottenere le spaziature di uso più comune:

d=0		righe sovrapposte
d=21	(*) ...	righe a contatto
d=24		9 righe per pollice
d=27		8 righe per pollice
d=31	(*)...	7 righe per pollice
d=36	(default)	6 righe per pollice
d=43	(*)...	5 righe per pollice
d=54		4 righe per pollice
d=72		3 righe per pollice
d=108		2 righe per pollice

I valori contrassegnati da (*) sono approssimati, non esistendo un numero intero di passi tale da creare la giusta distanza tra le righe; tuttavia l'errore si può considerare trascurabile. In particolare, per quanto riguarda la stampa con righe a contatto il valore di "d" è compreso tra 20 e 21, ma noi abbiamo scartato il valore inferiore, poiché crea una leggera, ma antiestetica sovrapposizione tra due righe adiacenti. Come vedremo più avanti, tale valore è molto importante, in quanto è quello utilizzato nella spaziatura della stampa ad alta risoluzione.

Analizziamo infine brevemente l'opzione che consente di selezionare il numero di righe per pagina. Come nel caso precedente, si inviano alla stampante i comandi:

OPEN3,4,3:PRINT #3,CHR\$(n):CLOSE3

dove "n" è un numero o una variabile il cui valore deve essere compreso tra 14 e 127. Esso rappresenta il numero di righe effettivamente stampate in una pagina quando nella MPS-802 è attivata la separazione di pagina. A tale numero vanno aggiunte 6 righe vuote (la separazione), distribuite tre all'inizio e tre alla fine di ogni pagina. Ricordiamo che per attivare la separazione di pagina è sufficiente inviare, tramite un file di stampa (indirizzi secondari 0-1-7), un CHR\$(147); per disabilitarla si usa invece CHR\$(19). Quest'ultimo carattere, se la separazione è attivata, fa avanzare la carta fino all'inizio della pagina successiva. Il valore di "n" all'accensione della stampante (default) è 60: infatti la lunghezza di pagina dei tabulati standard in modulo continuo è di 11 pollici, per cui stampando 6 righe per pollice ne abbiamo 66 per pagina, dalle quali vanno sottratte le 6 righe di separazione.

Il programma n. 1

All'accensione la MPS-802 viene inizializzata nel modo seguente:

- numero di righe per pagina (n) = 60 (lunghezza pagina = 66);
- righe per pollice = 6 (d=36);
- separazione di pagina disabilitata;
- stampa messaggi di errore disabilitata.

Per modificare questa "taratura" occorre inviare alla stampante, mediante l'apertura di opportuni file, determinati caratteri, che vengono interpretati alla stregua di comandi. Tale procedura è

Indubbiamente un po' macchinosa, per cui abbiamo pensato di proporvi un breve programma (listato n. 1) che reinizializza la stampante coi valori richiesti. Il programma gira su tutti i computer Commodore. Esso vi chiede innanzitutto quante righe per pollice volete stampare (per mantenere il default premete RETURN) e se desiderate la separazione tra le pagine; in caso positivo dovete introdurre il numero di righe per pagina (per tabulati da 11 pollici premete RETURN). Infine vi viene chiesto se gli eventuali messaggi di errore devono essere stampati. A questo punto viene restituito il controllo all'interprete BASIC, e la MPS-802 resta "programmata" nel modo selezionato finché non riceve ulteriori comandi oppure non viene reinizializzata o spenta. Sarà quindi sufficiente, ogni volta che dovete usare la stampante, caricare questo programmino nella memoria del vostro computer e dare il RUN: in pochi secondi la vostra MPS-802 verrà predisposta a stampare esattamente nel modo desiderato.

I caratteri personalizzati

Gli 8 byte che codificano un carattere programmabile vengono inviati, tramite un file con indirizzo secondario 5, alla stampante, la quale li memorizza in un'apposita area della sua memoria; in seguito, ogni volta che tramite un file di stampa (indirizzi secondari 0-1-7) viene inviato un CHR\$(254) il carattere personalizzato viene stampato. Abbiamo già accennato al fatto che tali caratteri non possono essere più di uno per riga. Per ovviare a questa limitazione è necessario far ricorso al carattere CHR\$(141) ("carriage return with no line feed" = vai a capo senza far avanzare la carta), che permette di stampare più righe sovrapposte ("overprinting"). Con tale tecnica, che peraltro può essere utilizzata anche per ottenere il "neretto" (facendo stampare più volte una certa scritta nello stesso punto), possiamo avere fino a 80 caratteri personalizzati sulla stessa riga. L'unico piccolo problema è la necessità di memorizzare in qualche modo il punto della riga in cui deve avvenire la stampa, dato che il CHR\$(141) fa "dimenticare" alla stam-

pante a quale colonna era arrivata. La soluzione più semplice consiste nell'utilizzare la funzione TAB del BASIC, usando come argomento una variabile contenente il numero della colonna in cui deve avvenire la stampa. Per chiarire ulteriormente le idee vi mostriamo una breve routine che stampa tre caratteri personalizzati (tre piccole clessidre) sulla stessa riga.

```
10 DATA 129, 195, 229, 249, 249, 229, 195, 129
20 DATA 129, 195, 167, 187, 167, 195, 129
30 DATA 129, 195, 167, 159, 159, 167, 195, 129
40 OPEN 5,4:OPEN 4,4
50 FOR I=0 TO 4 STEP 2
60 AS="":FOR J=1 TO 8:READ A$=A$+CHR$(A):NEXT J
70 PRINT # 5, A$
80 PRINT # 4, TAB (I) CHR$ (254) CHR$ (141):
90 NEXT I:PRINT # 4:CLOSE 4:CLOSE 5:END
```

Linee 10-30 : dati che formano i tre caratteri, ricavati nel modo illustrato sul manuale della stampante.

Linea 40: apertura del file con indirizzo secondario 5, che serve ad inviare alla stampante un carattere programmabile, il quale viene memorizzato nell'apposito buffer; apertura del file di stampa.

Linea 50: inizio del ciclo di stampa dei tre caratteri.

Linea 60: ciclo di costruzione di ogni carattere a partire dalle 8 colonne che lo formano: ogni colonna è composta da 8 bit e costituisce quindi un byte. Gli 8 byte vengono "concatenati" per formare la stringa AS.

Linea 70: la stringa AS, che contiene la codifica del carattere, viene inviata alla stampante tramite il file con indirizzo secondario 5.

Linea 80: viene stampato nella colonna opportuna (per mezzo della funzione TAB) il carattere CHR\$(254), seguito da CHR\$(141) e da un punto e virgola, il quale fa sì che non venga inviato alla stampante anche un CHR\$(13), con conseguente avanzamento della carta.

Linea 90: fine del ciclo di stampa, avanzamento della carta e chiusura del file. Ci auguriamo che questo esempio possa esservi utile nella gestione dei caratteri personalizzati. La routine è facilmente modificabile, per stampare una maggior quantità di caratteri programmabili su un'unica riga (aumentando il numero di istruzioni DATA e il numero di volte che il ciclo di stampa viene eseguito).

La stampa ad alta risoluzione

Il discorso della grafica ad alta risoluzione è strettamente legato a quello dei caratteri programmabili: infatti, per comandare i singoli punti della stampante dobbiamo comunque passare attraverso un lungo (e piuttosto noioso) lavoro di costruzione di caratteri personalizzati. In questo caso la distanza tra le righe deve essere di 21 passi, in modo che i caratteri siano a contatto. Il secondo programma che pubblichiamo costituisce un esempio di come si possano

sfruttare al meglio le qualità grafiche della MPS-802.

Il programma n. 2

Esiste in circolazione molto software che crea o permette di creare spettacolari immagini ad alta risoluzione sullo schermo del C64, per cui pensiamo di fare una gradita sorpresa a tutti coloro che possiedono tale computer ed una stampante MPS-802, presentando una routine in linguaggio macchina capace di copiare su carta le schermate hi-res del C64. Abbiamo strutturato tale routine in modo analogo a quella scritta per le altre stampanti Commodore e comparso su SUPERCOMMODORE n. 4 ("Stampare lo schermo ad alta risoluzione").

Il programma (listato n. 2) memorizza il linguaggio macchina all'interno del blocco di RAM che inizia da 49152 (\$C000), in modo che non interferisca con la memoria BASIC, e consente di scegliere le dimensioni della stampa. Esistono infatti ben tre grandezze possibili:

- 1 = normale: un punto sul video corrisponde a un punto della stampante;
- 2 = doppia larghezza e doppia altezza;
- 3 = tripla larghezza e doppia altezza.

Il valore selezionato viene memorizzato nel byte di indirizzo 2. Dato che il programma BASIC, dopo aver compiuto la sua funzione, si autodistrugge con un comando NEW, per cambiare in seguito la grandezza di stampa bisogna digitare POKE 2,g, dove g può valere 1,2 oppure 3.

La stampa dello schermo hi-res viene comandata con SYS50000, e può essere interrotta in qualsiasi momento premendo il tasto RUN/STOP. Tale SYS può venire inserita nel programma che crea l'immagine ad alta risoluzione e funziona soltanto se, al momento in cui viene eseguita, lo schermo contiene la pagina grafica. È tuttavia possibile effettuare la stampa di un disegno in memoria anche mentre questo non appare sul video, cioè quando lo schermo è in bassa risoluzione, a patto di conoscere quale area di memoria contenga la pagina grafica. Si deve allora effettuare una POKE nella locazione 900, assegnandole il valore del byte alto dell'indirizzo iniziale dello schermo hi-res: se, ad esempio, la pagina grafica inizia alla locazione 57344 (\$E000), si deve digitare POKE 900,57344/256. In questo caso la stampa viene attivata con SYS 50032.

Può capitare che non abbiate la più pallida idea riguardo alla posizione in memoria della pagina grafica. In tal caso non perdetevi d'animo: tenete presente che una schermata ad alta risoluzione occupa ben 8000 byte (quasi 8 Kbyte) e che ci sono solo 8 blocchi di 8 Kbyte nella memoria del vostro C64. Potete quindi procedere per tentativi, in quanto non sono molti i valori assegnabili alla locazione 900.



Vediamoli:

POKE 900,32 (inizio schermo hi-res = 32*256 = 8192 = \$2000)

POKE 900,64 (inizio = 64*256 = 16384 = \$4000)

POKE 900,96 (inizio = 96*256 = 24576 = \$6000)

POKE 900,128 (inizio = 128*256 = 32768 = \$8000)

POKE 900,160 (inizio = 160*256 = 40960 = \$A000)

POKE 900,224 (inizio = 224*256 = 57344 = \$E000)

Abbiamo escluso che la pagina grafica inizi da 0 (questo blocco non viene mai usato) oppure da 49152 (in questo caso cancellerebbe la nostra routine). Riassumendo, se avete uno schermo hi-res memorizzato che inizia (per esempio) da 16384 e lo volete stampare, digitate:

POKE900,64:SYS50032

Per concludere, vi diamo un'ultima informazione, di cui dovete tener conto nel selezionare le dimensioni della stampa. Sono sufficienti circa 4 minuti per una riproduzione a grandezza naturale (g=1); se però desiderate gli ingrandimenti (g=2 oppure g=3), sappiate che il tempo di stampa sale rispettivamente a 30 e a 45 minuti circa.

Programma 1

```
100 DATA108,72,54,43,36,31,27,24:rem 77
110 FORI=2TO9:READR(I):NEXT :rem 200
120 PRINT"{CLR}{RVS}INIZIALIZZA
    { 2 SPAZI}MPS-802" :rem 14
130 PRINT"{ 2 GIU'}ACCENDI LA STAMPANTE":
    PRINT"E PREMI UN TASTO." :rem 209
140 GETA$:IFA$=" "THEN140 :rem 77
150 OPEN3,4,3:OPEN4,4,4:OPEN6,4,6:OPEN8,4,
    0:OPEN10,4,10 :rem 152
160 PRINT#10 :rem 168
170 PRINT"{ 2 GIU'}RIGHE PER POLLICE
    { 2 SPAZI}6 { 4 SIN}";INPUTD :rem 220
180 IFD<2ORD>9THENPRINT"{ 4 SU}":GOTO170 :rem 133
190 PRINT#6,CHR$(R(D)) :rem 229
```

Programma 2

```
100 INPUT"{CLR}{WHT}GRANDEZZA (1-2-3)";WI
    $ :rem 227
110 IFVAL(WI$)<1ORVAL(WI$)>3THEN100 :rem 80
120 POKE2,VAL(WI$) :rem 24
130 I=50000 :rem 15
140 READA:IFA=256THEN170 :rem 157
150 PRINT"{CLR}MEMORIZZAZIONE LINGUAGGIO
    MACCHINA":POKE646,A :rem 205
160 POKEI,A:I=I+1:CH=CH+A:GOTO140 :rem 226
170 IFCH<>58183THENPRINT"{GIU'}{WHT}ERROR
    E NELLE ISTRUZIONI DATA":END:rem 213
180 PRINT"{GIU'}{WHT}SYS 50000 PER INIZIA
    RE LA STAMPA":NEW :rem 45
50000 DATA173,0,221,41,3,73,3,160 :rem 111
50008 DATA6,10,136,208,252,141,132,3 :rem 14
50016 DATA173,24,208,41,8,240,9,24 :rem 183
50024 DATA169,32,109,132,3,141,132,3 :rem 16
50032 DATA169,15,141,182,3,165,2,201 :rem 19
50040 DATA1,240,5,169,14,141,182,3 :rem 172
50048 DATA169,0,32,189,255,169,6,168 :rem 55
50056 DATA162,4,32,186,255,32,192,255 :rem 88
50064 DATA162,6,32,201,255,176,3,76 :rem 238
50072 DATA157,195,76,30,197,169,21,32 :rem 97
50080 DATA210,255,169,13,32,210,255,32 :rem 119
50088 DATA204,255,169,0,32,189,255,169 :rem 152
```

```
200 PRINT"{GIU'}PAGINE SEPARATE
    { 2 SPAZI}S { 4 SIN}";INPUTPS :rem 189
210 IFPS="N"THEN260 :rem 44
220 PRINT"{GIU'}RIGHE PER PAGINA "D*11-6"
    { 5 SIN}";INPUTR :rem 91
230 IFR<14ORR>127THENPRINT"{ 3 SU}":GOTO2
    20 :rem 156
240 PRINT#3,CHR$(R) :rem 73
250 PRINT#8,CHR$(147) :rem 153
260 PRINT"{GIU'}MESSAGGI ERRORE
    { 2 SPAZI}S { 4 SIN}";INPUTES :rem 206
270 IFES="N"THEN290 :rem 42
280 PRINT#4 :rem 126
290 CLOSE3:CLOSE4:CLOSE6:CLOSE8:CLOSE10 :rem 7
300 PRINT"{ 2 GIU'}{RVS} OK ":END :rem 67
```

```
50096 DATA5,168,162,4,32,186,255,32 :rem 248
50104 DATA192,255,169,0,32,189,255,169 :rem 147
50112 DATA4,170,160,0,32,186,255,32 :rem 223
50120 DATA192,255,162,4,32,201,255,169 :rem 127
50128 DATA13,32,210,255,32,204,255,169 :rem 125
50136 DATA0,141,142,3,24,173,132,3 :rem 167
50144 DATA105,30,141,143,3,169,40,141 :rem 64
50152 DATA192,3,165,2,201,1,240,5 :rem 119
50160 DATA169,2,141,157,3,173,142,3 :rem 231
50168 DATA133,251,173,143,3,133,252,169 :rem 183
50176 DATA25,141,197,3,169,1,141,212 :rem 30
50184 DATA3,169,8,141,202,3,162,5 :rem 134
50192 DATA32,201,255,165,2,201,1,208 :rem 15
50200 DATA6,32,34,197,76,100,196,32 :rem 235
50208 DATA34,197,174,157,3,224,1,208 :rem 36
50216 DATA5,41,15,76,52,196,41,240 :rem 187
50224 DATA74,74,74,74,168,162,4,169 :rem 3
50232 DATA0,141,162,3,169,1,141,167 :rem 225
50240 DATA3,169,3,141,172,3,152,45 :rem 179
50248 DATA167,3,240,9,173,162,3,13 :rem 188
```


50256 DATA172,3,141,162,3,14,167,3
:rem 181
50264 DATA14,172,3,14,172,3,202,208
:rem 225
50272 DATA229,173,162,3,32,210,255,206
:rem 126
50280 DATA202,3,240,3,76,19,196,169
:rem 244
50288 DATA13,32,210,255,32,204,255,162
:rem 125
50296 DATA4,32,201,255,174,212,3,202
:rem 20
50304 DATA240,8,169,32,32,210,255,76
:rem 30
50312 DATA127,196,173,182,3,32,210,255
:rem 128
50320 DATA166,2,224,3,208,5,169,14
:rem 182
50328 DATA32,210,255,169,254,32,210,255
:rem 179
50336 DATA169,141,32,210,255,32,204,255
:rem 176
50344 DATA173,212,3,24,101,2,141,212
:rem 6
50352 DATA3,32,234,255,32,225,255,208
:rem 78
50360 DATA3,76,2,197,198,252,56,165
:rem 1
50368 DATA251,233,64,133,251,176,2,198
:rem 143
50376 DATA252,206,197,3,240,3,76,9
:rem 199
50384 DATA196,162,4,32,201,255,169,13
:rem 87
50392 DATA32,210,255,32,204,255,165,2
:rem 74
50400 DATA201,1,240,8,206,157,3,240
:rem 215

50408 DATA3,76,245,195,24,173,142,3
:rem 244
50416 DATA105,8,141,142,3,144,3,238
:rem 229
50424 DATA143,3,206,192,3,240,3,76
:rem 181
50432 DATA234,195,162,4,32,201,255,169
:rem 133
50440 DATA13,32,210,255,32,204,255,162
:rem 115
50448 DATA6,32,201,255,169,36,32,210
:rem 30
50456 DATA255,169,13,32,210,255,32,231
:rem 129
50464 DATA255,96,120,162,52,134,1,172
:rem 82
50472 DATA202,3,136,177,251,162,55,134
:rem 131
50480 DATA1,88,96,256
:rem 76

ERRATA CORRIGE

Nel programma
"Paratrooper per
C64" pubblicato sul
numero 5, il valore
della variabile MB
in linea 160
va modificato da
53246 in 53264.
Ci scusiamo
con i lettori.

**Per ricevere le riviste arretrate
di SUPERCOMMODORE con cassetta
compilate e spedite il coupon qui sotto!**

SUPERCOMMODORE

CEDOLA DI ORDINAZIONE RIVISTE ARRETRATE CON CASSETTA ALLEGATA
da compilare e spedire in busta chiusa a
J. soft - Viale Restelli, 5 - 20124 Milano - Tel. 6888228

Vogliate inviarmi i numeri

di SUPERCOMMODORE con cassetta Anno

Al prezzo di L. 15.000 cad.

☐ Contanti allegati ☐ Assegno allegato n°

☐ Ho spedito l'importo a mezzo vaglia postale

☐ Ho versato l'importo sul ccp. n° 19445204 intestato a J. soft - Milano

☐ Pagherò in contantesse al postino al ricevimento

BUONO D'ORDINE PER RIVISTE ARRETRATE

Anche se l'ordine riguardasse la sola cassetta, questa
verrà comunque inviata insieme alla rivista al prezzo in-
dicato.

Nome

Cognome

Via

Città C.A.P. Prov.

Se richiesta fattura:

Cod. F. e P. Iva

Data

Firma

Per i residenti all'estero - pagamento anticipato (vaglia o versamento su ns. ccp)

OK-LIST

di E. Sarcina

Sappiamo tutti quanto sia difficile digitare correttamente un programma (ovviamente, se non si dispone della cassetta allegata a SUPERCOMMODORE!). Errori di battitura anche banali possono provocare il malfunzionamento del programma stesso, o anche il blocco completo del sistema, obbligandoci a rifare il lavoro.

Ma il punto di forza dei calcolatori è che essi eccellono nei compiti noiosi e che richiedono precisione. Scoprire un errore nella battitura di un listato è appunto un compito noioso, quindi perché non sfruttare il tuo computer per controllare i listati da te digitati?

È proprio questo lo scopo di OK-LIST, il programma che ti permetterà di risparmiare tempo e fatica. Annidato all'interno del tuo computer, OK-LIST controlla automaticamente ogni riga battuta, mostrandoti un numero in cima allo schermo ogni volta che premi il tasto RETURN. Un numero corrispondente nel listato del programma pubblicato ti permette il confronto, ed uno sguardo è sufficiente per avere la conferma di aver digitato la linea correttamente.

OK-LIST è un breve programma in linguaggio macchina residente nel buffer del registratore. Esso rimane in tale zona finché spengi il computer o usi l'unità a cassetta.

Come digitare OK-LIST

Vi sono due versioni di OK-LIST, una per C64 e VIC 20 e l'altra per C16 e Plus/4. Digita la versione adatta al tuo

computer, facendo attenzione a non commettere errori (per fortuna il listato è molto breve!). Dopo aver finito salvato su disco o nastro, per lo meno un paio di volte, prima di mandarlo in esecuzione con il comando RUN, perché il programma BASIC, dopo aver memorizzato la routine in linguaggio macchina (le istruzioni DATA), si autocancella.

Come si usa OK-LIST

A questo punto puoi iniziare a battere il tuo primo listato "a prova di errore". Se non hai OK-LIST in memoria, caricalo e attivalo con RUN.

Digita quindi la prima linea di programma: come puoi vedere, in fondo ad essa c'è la parola "rem" seguita da un numero, che si chiama checksum.

NON COPIARE QUESTA PARTE DELLA LINEA! Ora premi il tasto RETURN, come al solito. In alto a sinistra apparirà il checksum; confrontalo con quello che vedi scritto sulla rivista dopo "rem". Se i due numeri sono uguali, vuol dire che la linea è stata digitata correttamente, altrimenti controllala, perché evidentemente hai commesso un errore. Facile, no?

OK-LIST non è pignolo per quel che riguarda gli spazi bianchi, dato che non fa caso alla loro presenza. Ciò torna a tuo vantaggio, dal momento che la spaziatura non è, in genere, importante. Ma all'interno delle istruzioni PRINT gli spazi spesso si rivelano necessari, quindi presta attenzione quando digiti questo tipo di istruzione. Un'altra cosa a cui devi badare è che usando le forme abbreviate dei comandi (ad esempio ? per PRINT) il checksum visualizzato non coinciderà con quello presente sul listato. Ciò è tuttavia risolvibile, pre-

mendo il tasto RETURN su una linea dopo averla listata: le abbreviazioni saranno ricodificate nella loro forma intera e quindi OK-LIST sarà in grado di svolgere perfettamente la sua funzione. In questo modo, inoltre, avrai la possibilità di controllare interi gruppi di righe in una volta sola.

Ovviamente, nessuno è perfetto, ed anche OK-LIST ha qualche limite: se per caso ti capitasse (ma è un errore poco frequente) di "anagrammare" durante la digitazione una linea (per esempio invece di battere 10 PRINT "VAI", digiti per errore 10 PRINT "VIA"), OK-LIST non se ne accorgerebbe. Inoltre la presenza del carattere SHIFT-SPAZIO modifica il numero di checksum rispetto al carattere SPAZIO normale, anche se la differenza non risulta visibile sul video del tuo computer.

Se intendi sospendere il lavoro di digitazione — o lo hai terminato — puoi salvare su nastro o disco il tuo programma: ricordati comunque di premere RUN/STOP e RESTORE su VIC 20 e C64, oppure RUN/STOP e RESET su C16 e Plus/4 prima di dare il comando SAVE. Al momento di riprendere il lavoro, tuttavia, non potrai più utilizzare OK-LIST, dal momento che qualsiasi operazione su cassetta lo distrugge.

Se vuoi disabilitare OK-LIST in qualunque momento, senza perdere il tuo programma in memoria, premi RUN/STOP e RESTORE su VIC 20 e C64 oppure RUN/STOP e RESET su C16 e Plus/4. Per riattivarlo, batti:

SYS 886
Buon lavoro!

OK-LIST

versione per Vic 20 e C64

```
10 PRINT{CLR}ATTENDERE PREGO..
  "
11 FORI=886TO1018:READA:CK=CK+A
  :POKEI,A:NEXT
12 IFCK<>14714THENPRINT"[GIU']E
  RRORE NELLE ISTRUZIONI DATA"
  :END
13 SYS886:PRINT"[CLR]{ 2 GIU' }
  OK-LIST ATTIVATO.":NEW
14 DATA173,034,003,201,150,208
15 DATA001,096,141,151,003,173
16 DATA035,003,141,152,003,169
```

```
17 DATA150,141,034,003,169,003
18 DATA141,035,003,169,000,133
19 DATA006,096,032,232,235,133
```

```
20 DATA003,134,004,132,005,008
21 DATA201,013,240,017,201,032
22 DATA240,005,024,101,006,133
23 DATA006,165,003,166,004,164
24 DATA005,040,096,169,013,032
25 DATA210,255,165,205,141,251
```

```
26 DATA003,206,251,003,169,000
27 DATA133,207,169,019,032,210
```

```
28 DATA255,169,018,032,210,255
29 DATA169,058,032,210,255,166
30 DATA006,169,000,133,006,172
31 DATA151,003,192,232,208,006
32 DATA032,095,164,076,235,003
33 DATA032,007,208,169,032,032
34 DATA210,255,032,210,255,173
```

```
35 DATA251,003,133,205,076,173
36 DATA003
```

OK-LIST

versione per C16 e Plus/4

```
10 PRINT{CLR}ATTENDERE PREGO..
  "
11 FORI=886TO1018:READA:CK=CK+A
  :POKEI,A:NEXT
12 IF CK<>17539 THEN PRINT"
  [GIU']ERRORE NELLE ISTRUZION
  I DATA":END
13 SYS886:PRINT"[CLR]{ 2 GIU' }
  OK-LIST ATTIVATO.":NEW
```

```
14 DATA173,036,003,201,150,208
15 DATA001,096,141,151,003,173
16 DATA037,003,141,152,003,169
17 DATA150,141,036,003,169,003
```

```
18 DATA141,037,003,169,000,133
19 DATA254,096,032,087,241,133
20 DATA251,134,252,132,253,008
21 DATA201,013,240,017,201,032
22 DATA240,005,024,101,254,133
```

```
23 DATA254,165,251,166,252,164
24 DATA253,040,096,169,013,032
25 DATA210,255,165,214,141,251
26 DATA003,206,251,003,169,000
27 DATA133,216,169,019,032,210
```

```
28 DATA255,169,018,032,210,255
29 DATA169,058,032,210,255,166
30 DATA254,169,000,133,254,172
```

```
31 DATA151,003,192,087,208,006
32 DATA032,205,189,076,235,003
33 DATA032,205,221,169,032,032
34 DATA210,255,032,210,255,173
35 DATA251,003,133,214,076,173
36 DATA003
```




NO ZAP: un programma anti black-out

di J. Blake Lambert
trad. ed adatt.
di M. Cristuib Grizzi

Questa breve ed utile routine salva automaticamente e periodicamente su disco le versioni aggiornate del programma BASIC sul quale state lavorando, costituendo un aiuto prezioso per chi abita in zone in cui si verificano frequenti black-out elettrici. La versione per VIC 20 funziona sia con che senza espansioni di memoria.

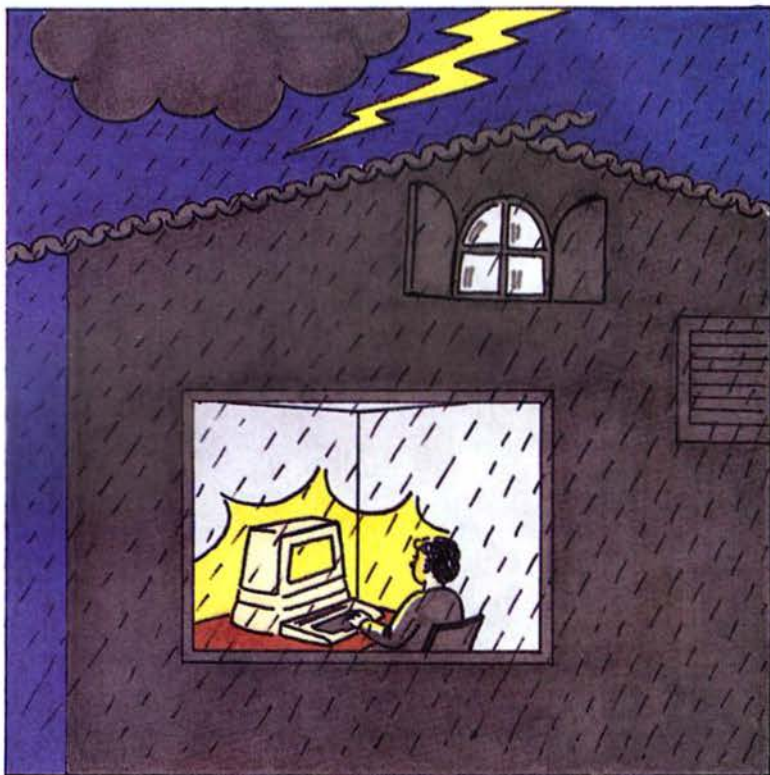
Se vi siete già imbattuti, lavorando sul calcolatore, in un black-out elettrico, in un grosso sbalzo di tensione, o se soltanto qualcuno è inciampato nel cavo di alimentazione del computer, provocando la perdita irreparabile del programma su cui stavate lavorando da quattro ore filate, apprezzerete senz'altro l'utilità di questo programma.

È facile infatti raccomandare a tutti di effettuare periodici salvataggi dei programmi su cui si sta lavorando: in realtà, quando le idee scorrono veloci in mente è altrettanto facile dimenticarsene o rimandare progressivamente il salvataggio. NO ZAP non solo vi ricorda questa incombenza, ma se ne fa addirittura carico, automaticamente e periodicamente.

Una volta mandato in esecuzione NO ZAP, ed inserito un nome di file, il programma salverà ogni dieci minuti la versione del programma che avete in memoria, aggiornandone il nome. Non dovrete compiere alcuna operazione particolare: tutto avverrà automaticamente.

Ogni dieci minuti NO ZAP attende che abbiate finito di digitare la linea su cui state lavorando e appena premete RETURN, per inserirla in memoria, salva tutto il programma su disco.

NO ZAP funziona anche con alcune utility o programmi di aiuto per la digitazione dei listati: funziona ad esempio insieme a OK-LIST, ma non con SPEED-SCRIPT o MLX (dal momento che MLX è un programma BASIC, NO ZAP farebbe un back-up di MLX stesso, invece che del programma in linguaggio macchina che state digitando...).



NO ZAP funziona anche insieme al DOS 5.1 ed a molti tipi di Wedge.

NO ZAP tiene conto delle dimensioni del programma sul quale state lavorando ed aggiunge automaticamente un numero progressivo di versione all'inizio del nome del file usato per il salvataggio. Sono disponibili fino a 100 versioni diverse (100 versioni * 10 minuti = 17 ore di lavoro continuato) numerate da 01 a 100, mentre al raggiungimento della 101-esima il conteggio riparte da 00. Esistono solo due limiti non valicabili da NO ZAP: lo spazio libero sul disco e lo spazio libero sulla directory; se non ci sono blocchi liberi a sufficienza, il pro-

gramma non verrà salvato e si ricordi che la directory può contenere al massimo 144 nomi di file.

Come usare NO ZAP

Il programma, una volta caricato e mandato in esecuzione, inserisce tramite istruzioni POKE una routine in linguaggio macchina a partire dalla cima corrente della memoria BASIC e la protegge dalle variabili BASIC. NO ZAP usa inoltre le locazioni di memoria da 739 a 767, che naturalmente non dovranno essere usate da altri programmi. Dopo avere dato il RUN a NO ZAP vi verrà chiesto:

FILENAME?

Digitate un nome di file senza virgolette, composto da un numero di caratteri compreso tra zero e quattordici, e premete RETURN (non cercate di usare un nome di file più lungo di quattordici caratteri, perché ciò provocherebbe il blocco del computer!). Non dovete aggiungere al nome il numero della versione, perché NO ZAP provvede automaticamente ad aggiungerlo per conto vostro. Infine digitate NEW e premete RETURN.

Da questo punto in poi dimenticatevi pure di NO ZAP: esso agirà autonomamente e voi potrete tranquillamente salvare, caricare, eseguire e verificare, come sempre, i programmi sui quali state lavorando. La prima volta che NO ZAP esegue un salvataggio usa un numero di versione 01; per esempio, se usate come nome di file PIPPO, la prima versione sarà 01PIPO, la seconda 02PIPO e così via. NO ZAP visualizza lo stato del disco dopo il salvataggio, ma non ripete l'operazione in caso di errore.

Salvataggi forzati ed interruttore

Occasionalmente, potreste voler salvare una nuova versione prima del prossimo salvataggio effettuato da NO ZAP. Oppure potreste voler disattivare temporaneamente NO ZAP, perché andate a colazione o comunque interrompete il lavoro per riprenderlo dopo un certo tempo. Per ottenere questi scopi usate rispettivamente le istruzioni (in modo diretto):

SYS 739 (salvataggio forzato)
SYS 745 (interruttore acceso/spento)

La SYS 739 incrementa il numero di versione e salva il programma: NO ZAP ripristina a zero il suo timer ed il salvataggio successivo avverrà dopo dieci minuti dalla SYS.

La SYS 745 funziona come un interruttore: se NO ZAP è attivo, con questa SYS viene disattivato, mentre se è disattivato, questa SYS provvede a riattivarlo (come se gli fosse stato dato il RUN per la prima volta).

Alcuni trucchetti

NO ZAP è stato scritto in modo da non interferire con la vostra programmazione: RUN/STOP+RESTORE quindi non lo disattiva. L'unico modo per cancellare NO ZAP dalla memoria del computer è spegnere quest'ultimo, oppure eseguire una SYS64738 sul C64, o una SYS64802 sul VIC 20.

Se, una sera, lavorando ad un programma decidete di andare a dormire una volta giunti alla versione 04PIPO, il giorno dopo potrete utilizzare ancora PIPPO come nome del file, togliere il disco dal drive e dare quattro volte la SYS 739; quindi spegnete e riaccendete il drive: così facendo, NO ZAP conti-

nuerà a salvare, partendo dalla versione 05PIPO.

Dal momento che i numeri di versione sono posti all'inizio del nome del file, potete listare sullo schermo tutte le versioni fino ad ora salvate, usando:

LOAD"\$:??PIPO",8 [RETURN]
LIST [RETURN]

oppure, se usate il DOS, con

@:\$??PIPO

Una volta finito di lavorare sul programma potreste voler cancellare dal disco tutte le versioni intermedie registrate da NO ZAP. Ciò si ottiene con:

OPEN15,8,15,"S:\$??PIPO":
CLOSE15

oppure, se usate il DOS, con:

@:\$??PIPO

Assicuratevi di aver prima salvato la versione definitiva su un altro disco, oppure sullo stesso disco, ma con un altro nome.

Come lavora NO ZAP

NO ZAP trae vantaggio dal fatto che molte delle routine dell'interprete BASIC e del sistema operativo sono vettorizzate. Un vettore è una specie di tavola che dice al computer la locazione di una routine. Dal momento che il vettore è in RAM, esso può essere modificato per puntare ad un'altra routine appositamente approntata. Un programma che usa questa tecnica è chiamato

Disassemblato della routine (per C64)

```

.. 9E00 AD EC 02 LDA $02EC
.. 9E03 49 01 EOR $01
.. 9E05 8D EC 02 STA $02EC
.. 9E08 D0 0D BNE $9E17
.. 9E0A A9 83 LDA $83
.. 9E0C 8D 02 03 STA $0302
.. 9E0F A9 A4 LDA $A4
.. 9E11 8D 03 03 STA $0303
.. 9E14 4C 83 A4 JMP $A483
.. 9E17 A9 30 LDA $30
.. 9E19 8D EF 02 STA $02EF
.. 9E1C 8D F0 02 STA $02F0
.. 9E1F A9 D2 LDA $D2
.. 9E21 A0 9E LDY $9E
.. 9E23 20 1E AB JSR $AB1E
.. 9E26 20 F9 AB JSR $ABF9
.. 9E29 A0 02 LDY $02
.. 9E2B B9 FE 01 LDA $01FE,Y
.. 9E2E 99 EF 02 STA $02EF,Y
.. 9E31 F0 03 BEQ $9E36
.. 9E33 C8 INY
.. 9E34 D0 F5 BNE $9E2B
.. 9E36 8C ED 02 STY $02ED
.. 9E39 A2 53 LDX $53
.. 9E3B A0 9E LDY $9E
.. 9E3D 8E 02 03 STX $0302
.. 9E40 8C 03 03 STY $0303
.. 9E43 A9 01 LDA $01

```

```

.. 9E45 8D EE 02 STA $02EE
.. 9E48 A9 00 LDA $00
.. 9E4A 8D 08 DC STA $DC08
.. 9E4D 8D 09 DC STA $DC09
.. 9E50 8D 0A DC STA $DC0A
.. 9E53 AD 0A DC LDA $DC0A
.. 9E56 29 F0 AND $F0
.. 9E58 F0 75 BEQ $9ECF
.. 9E5A CE EE 02 DEC $02EE
.. 9E5D D0 B9 BNE $9E48
.. 9E5F EE F0 02 INC $02F0
.. 9E62 AD F0 02 LDA $02F0
.. 9E65 C9 3A CMP $3A
.. 9E67 D0 14 BNE $9E7D
.. 9E69 A9 30 LDA $30
.. 9E6B 8D F0 02 STA $02F0
.. 9E6E EE EF 02 INC $02EF
.. 9E71 AD EF 02 LDA $02EF
.. 9E74 C9 3A CMP $3A
.. 9E76 D0 05 BNE $9E7D
.. 9E78 A9 30 LDA $30
.. 9E7A 8D EF 02 STA $02EF
.. 9E7D AD ED 02 LDA $02ED
.. 9E80 A2 EF LDX $EF
.. 9E82 A0 02 LDY $02
.. 9E84 20 BD FF JSR $FFBD
.. 9E87 A9 01 LDA $01
.. 9E89 A2 08 LDX $08
.. 9E8B A0 00 LDY $00
.. 9E8D 20 BA FF JSR $FFBA
.. 9E90 A9 2B LDA $2B
.. 9E92 A6 2D LDX $2D
.. 9E94 A4 2E LDY $2E
.. 9E96 20 D8 FF JSR $FFD8
.. 9E99 A9 8D LDA $8D
.. 9E9B 20 D2 FF JSR $FFD2
.. 9E9E A9 00 LDA $00
.. 9EA0 20 BD FF JSR $FFBD
.. 9EA3 A9 0F LDA $0F
.. 9EA5 A2 08 LDX $08
.. 9EA7 A0 0F LDY $0F
.. 9EA9 20 BA FF JSR $FFBA
.. 9EAC 20 C0 FF JSR $FFC0
.. 9EAF A2 0F LDX $0F
.. 9EB1 20 C6 FF JSR $FFC6
.. 9EB4 20 CF FF JSR $FFCF
.. 9EB7 C9 0D CMP $0D
.. 9EB9 F0 06 BEQ $9EC1
.. 9EBB 20 D2 FF JSR $FFD2
.. 9EBE 38 SEC
.. 9EBF B0 F3 BCS $9EB4
.. 9EC1 20 D2 FF JSR $FFD2
.. 9EC4 A9 0F LDA $0F
.. 9EC6 20 C3 FF JSR $FFC3
.. 9EC9 20 CC FF JSR $FFCC
.. 9ECC 4C 43 9E JMP $9E43
.. 9ECF 4C 83 A4 JMP $A483
.. 9ED2 12
.. 9ED3 4E 4F 5A LSR $5A4F
.. 9ED6 41 50 EOR ($50,X)
.. 9ED8 92
.. 9ED9 20 42 59 JSR $5942
.. 9EDC 20 42 4C JSR $5942
.. 9EDF 41 4B EOR ($4B,X)
.. 9EE1 45 20 EOR $20
.. 9EE3 4C 41 4D JMP $4D41
.. 9EE6 42
.. 9EE7 45 52 EOR $52
.. 9EE9 54
.. 9EEA D0 46 49 ORA $4946
.. 9EED 4C 45 4E JMP $4E45
.. 9EF0 41 4D EOR ($4D,X)
.. 9EF2 45 00 EOR $00

```


"Wedge". NO ZAP costruisce una "deviazione" nel ciclo principale dell'interprete BASIC, in modo che il controllo venga passato a NO ZAP stesso ad ogni pressione del tasto RETURN. Quando si dà il RUN a NO ZAP, e si inserisce il nome del file, quest'ultimo viene memorizzato in un apposito buffer, subito dopo il numero di versione corrente. Il vettore nelle locazioni \$302 e \$303, che punta al ciclo principale BASIC, viene alterato ed uno dei timer interni

al computer è posto a zero. NO ZAP legge il valore di questo timer ogni volta che viene premuto il tasto RETURN (e quindi inserita una linea BASIC): la versione per C64 utilizza uno dei due TOD (Time Of Day) disponibili nel computer nelle locazioni \$DC08-\$DC0A, mentre la versione per Vic 20 legge il "jilly clock" nelle locazioni \$A0-\$A1. Le operazioni di input/output su nastro possono quindi falsare gli intervalli di tempo tra un salvataggio e l'altro nella versio-

ne per VIC 20. Se il timer non ha ancora contato l'equivalente di dieci minuti, NO ZAP ripassa il controllo all'interprete BASIC (in \$A483 nel C64, \$C483 nel VIC 20). Dal momento che NO ZAP modifica il vettore principale nelle locazioni \$302 e \$303, non è compatibile con altri "Wedge" che utilizzano le stesse locazioni. A parte viene riportato il disassemblato di NO ZAP versione per C64, ottenuto tramite il SUPERMONITOR.

No Zap versione per VIC 20

```

2 POKE56,PEEK(56)-1:CLR:I=256*PEEK(56)+PE
  EK(55):S=I                                     :rem 216
4 DEFFNH(X)=INT(X/256)                           :rem 37
6 DEFFNL(X)=X-FNH(X)*256                         :rem 156
8 DATA 169,141,141,161,0,96,76                 :rem 202
10 FORJ=739TO745:READK:POKEJ,K:NEXT              :rem 249
12 POKE746,PEEK(55):POKE747,PEEK(56)            :rem 87
14 PRINT"SYS739=SAVATAGGIO FORZATO":PRIN
  T"SYS745=INTERRUTTORE."                       :rem 26
16 READA:B=B+A:IFA=256THEN102                   :rem 209
18 POKEI,A:I=I+1:GOTO16                          :rem 141
20 DATA 173,236,2,73,1,141                     :rem 241
22 DATA 236,2,208,13,169,131                   :rem 90
24 DATA 141,2,3,169,196,141                     :rem 45
26 DATA 3,3,76,131,196,169                     :rem 6
28 DATA 48,141,239,2,141,240                   :rem 94
30 DATA 2,169,7,160,7,32                        :rem 149
32 DATA 30,203,32,249,203,160                   :rem 131
34 DATA 2,185,254,1,153,239                     :rem 48
36 DATA 2,240,3,200,208,245                     :rem 33
38 DATA 140,237,2,162,7,160                     :rem 43
40 DATA 7,142,2,3,140,3                         :rem 85
42 DATA 3,169,1,141,238,2:REM CAMBIARE IL
  DATO '1' PER UN INTERVALLO PIU' LUNGO         :rem 119
44 DATA 169,0,133,160,133,161                   :rem 139
46 DATA 133,162,165,161,56,201                   :rem 194
48 DATA 141,144,116,206,238,2                   :rem 144
50 DATA 208,236,238,240,2,173                   :rem 144
52 DATA 240,2,201,58,208,20                     :rem 33
54 DATA 169,48,141,240,2,238                   :rem 102
56 DATA 239,2,173,239,2,201                   :rem 47
58 DATA 58,208,5,169,48,141                   :rem 65
60 DATA 239,2,173,237,2,162                   :rem 46
62 DATA 239,160,2,32,189,255                   :rem 104
64 DATA 169,1,162,8,160,0                     :rem 201
66 DATA 32,186,255,169,43,166                   :rem 166
68 DATA 45,164,46,32,216,255                   :rem 108
70 DATA 169,141,32,210,255,169                 :rem 199
72 DATA 0,32,189,255,169,15                   :rem 56
74 DATA 162,8,168,32,186,255                   :rem 113
76 DATA 32,192,255,162,15,32                   :rem 100
78 DATA 198,255,32,207,255,201                 :rem 208
80 DATA 13,240,6,32,210,255                     :rem 34
82 DATA 56,176,243,32,210,255                   :rem 150
84 DATA 169,15,32,195,255,32                   :rem 109
86 DATA 204,255,76,7,7,76                     :rem 222
88 DATA 131,196,18,78,79,90                     :rem 76
90 DATA 65,80,146,32,66,89                     :rem 15
92 DATA 32,66,76,65,75,69                     :rem 229
94 DATA 32,76,65,77,66,69                     :rem 233
96 DATA 82,84,13,70,73,76                     :rem 221
98 DATA 69,78,65,77,69,0,256                   :rem 134

```

```

102 IFB<>28586THENPRINT"ERRORE NELLE ISTR
  UZIONI DATA!":END                             :rem 150
104 POKES+32,FNL(S+206):POKES+34,FNH(S+20
  6)                                                :rem 215
106 POKES+58,FNL(S+80):POKES+60,FNH(S+80)       :rem 128
108 POKES+201,FNL(S+67):POKES+202,FNH(S+6
  7)                                                :rem 224
110 SYS745                                         :rem 49
READY.                                           :rem 173

```

No Zap versione per C64

```

2 POKE56,PEEK(56)-1:CLR:I=256*PEEK(56)+PE
  EK(55):S=I                                     :rem 216
4 DEFFNH(X)=INT(X/256)                           :rem 37
6 DEFFNL(X)=X-FNH(X)*256                         :rem 156
8 DATA 169,16,141,10,220,96,76                 :rem 200
10 FORJ=739TO745:READK:POKEJ,K:NEXT              :rem 249
12 POKE746,PEEK(55):POKE747,PEEK(56)            :rem 87
14 PRINT"SYS739=SAVATAGGIO FORZATO":PRIN
  T"SYS745=INTERRUTTORE."                       :rem 56
16 READA:B=B+A:IFA=256THEN102                   :rem 209
18 POKEI,A:I=I+1:GOTO16                          :rem 141
20 DATA 173,236,2,73,1,141                     :rem 241
22 DATA 236,2,208,13,169,131                   :rem 90
24 DATA 141,2,3,169,164,141                     :rem 40
26 DATA 3,3,76,131,164,169                     :rem 1
28 DATA 48,141,239,2,141,240                   :rem 94
30 DATA 2,169,7,160,7,32                        :rem 149
32 DATA 30,171,32,249,171,160                   :rem 139
34 DATA 2,185,254,1,153,239                     :rem 48
36 DATA 2,240,3,200,208,245                     :rem 33
38 DATA 140,237,2,162,7,160                     :rem 43
40 DATA 7,142,2,3,140,3                         :rem 85
42 DATA 3,169,1,141,238,2:REM CAMBIARE IL
  DATO '1' PER UN INTERVALLO PIU' LUNGO         :rem 119
44 DATA 169,0,141,8,220,141                     :rem 38
46 DATA 9,220,141,10,220,173                   :rem 83
48 DATA 10,220,41,240,240,117                   :rem 129
50 DATA 206,238,2,208,233,238                   :rem 145
52 DATA 240,2,173,240,2,201                   :rem 27
54 DATA 58,208,20,169,48,141                   :rem 106
56 DATA 240,2,238,239,2,173                   :rem 49
58 DATA 239,2,201,58,208,5                     :rem 2
60 DATA 169,48,141,239,2,173                   :rem 105
62 DATA 237,2,162,239,160,2                     :rem 44
64 DATA 32,189,255,169,1,162                   :rem 109
66 DATA 8,160,0,32,186,255                     :rem 1
68 DATA 169,43,166,45,164,46                   :rem 118
70 DATA 32,216,255,169,141,32                   :rem 146
72 DATA 210,255,169,0,32,189                   :rem 101
74 DATA 255,169,15,162,8,160                   :rem 107

```


LOAD



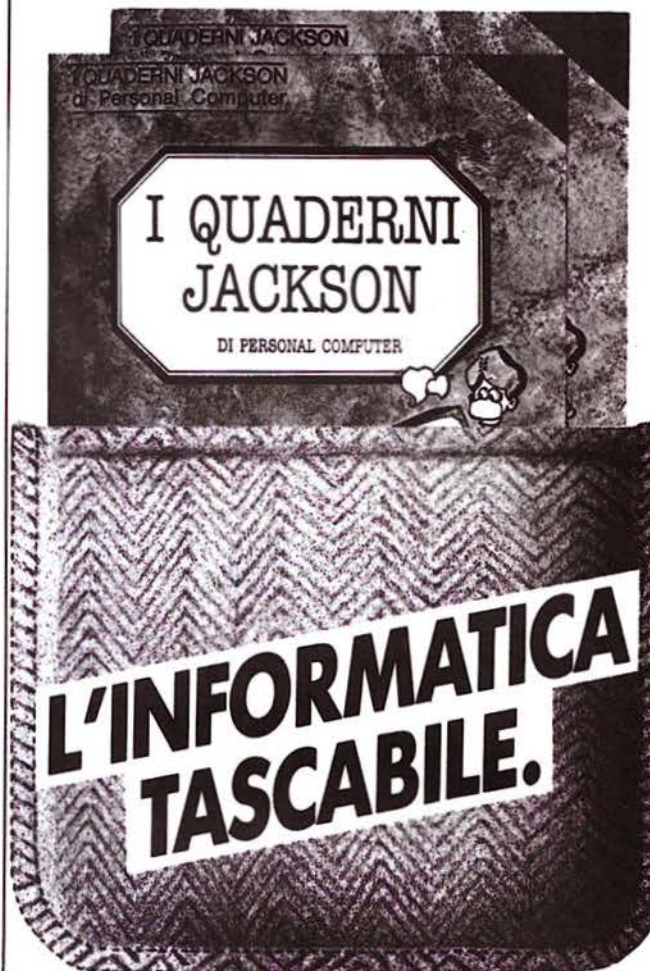
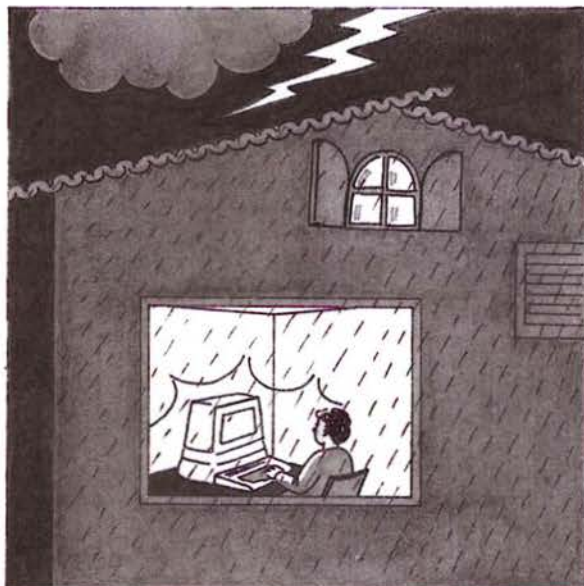
No Zap

```
76 DATA 15,32,186,255,32,192 :rem 106
78 DATA 255,162,15,32,198,255 :rem 163
80 DATA 32,207,255,201,13,240 :rem 133
82 DATA 6,32,210,255,56,176 :rem 51
84 DATA 243,32,210,255,169,15 :rem 149
```

```
86 DATA 32,195,255,32,204,255 :rem 155
88 DATA 76,7,7,76,131,164 :rem 222
90 DATA 18,78,79,90,65,80 :rem 227
92 DATA 146,32,66,89,32,66 :rem 15
94 DATA 76,65,75,69,32,76 :rem 232
96 DATA 65,77,66,69,82,84 :rem 239
98 DATA 13,70,73,76,69,78 :rem 231
100 DATA 65,77,69,0,256 :rem 112
102 IFB<>28715THENPRINT"ERRORE NELLE ISTR
    UZIONI DATA!":END :rem 144
104 POKES+32,FNL(S+210):POKES+34,FNH(S+21
    0) :rem 205
```

```
106 POKES+58,FNL(S+83):POKES+60,FNH(S+83)
    :rem 134
108 POKES+205,FNL(S+67):POKES+206,FNH(S+6
    7) :rem 232
110 SYS745 :rem 49
```

```
READY. :rem 173
```



Quaderni Jackson:
l'informatica a tutti i livelli, in
una collana aperta, pratica,
essenziale, aggiornata.

Tutto quello che è
importante sapere sui
computer, la programmazione,
i linguaggi, il software, le
applicazioni e i nuovi sviluppi
dell'informatica.

Ogni mese, 2 volumi.

Volumi già pubblicati:

Gianni Giaccagnini
"Vivere col Personal Computer"
Paolo Bozzola
"Dentro e fuori la scatola"

Enrico Odetti
"Ed è subito BASIC Vol. I"
"Ed è subito BASIC Vol. II"
Paolo Capobuzzi
e Marco Giacobazzi
"A ciascuno il suo Personal"
Fulvio Francesconi
e Fernando Paterlini
"To do or not to do"
Gianni Giaccagnini
"Strutturare il software"
Enrico Odetti
"Dizionario informatico"

In edicola,
a sole lire 6.000



GRUPPO
EDITORIA
JACKSON

SAN FRANCISCO LONDRA MILANO



3D color maze per C16

di A. Motta

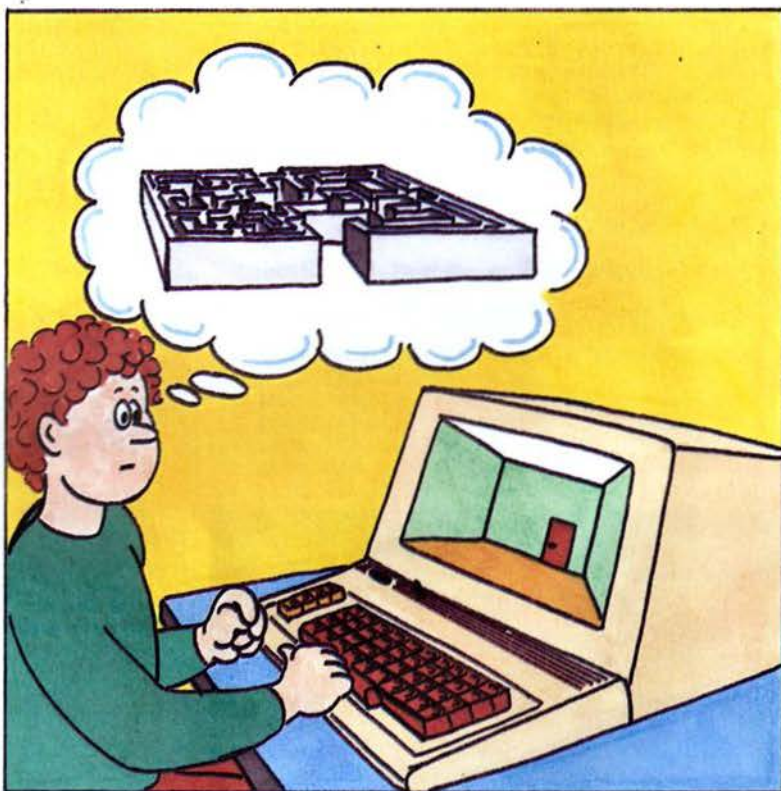
Il programma è la conversione di quello apparso sulla rivista ZX Computing di feb/mar '84 per il personal ZX81 e Plus/4.

Pur mantenendo inalterata l'impostazione di gioco, è stato in gran parte riscritto, a seguito delle differenze esistenti fra le due macchine, e sono stati aggiunti miglioramenti sia dal punto di vista grafico, vista la possibilità di avere a disposizione i colori, sia dal punto di vista del sonoro.

Sul labirinto c'è poco da dire: dovete solamente uscirne e, come per ogni labirinto che si rispetti, l'impresa non sarà facile.

Per muovervi dovete usare i tasti con i cursori (alto, sinistro e destro), che rappresentano le relative direzioni. Rispetto alla versione originaria sono stati creati due livelli di gioco: principiante ed esperto.

Nel primo avete a disposizione un tempo massimo di cinque minuti ed a video viene mostrata in continuazione la distanza in linea d'aria fra il punto in cui vi trovate e l'uscita; nel secondo il tempo a disposizione è ridotto a tre minuti e le informazioni relative alla distanza fra voi e l'uscita appaiono saltuariamente.



Elenco delle variabili utilizzate

A\$ (27) matrice contenente la griglia del labirinto. Composta dalla lettera "A", rappresentante il muro, e dalla lettera "B", rappresentante il passaggio.

P(8,2) matrice contenente le 4 direzioni (N,S,E,O). Entrambe in doppio, per semplificarne il controllo.
U e V coordinate del giocatore all'interno del labirinto.
X e Y utilizzate per la mossa corrente all'interno del labirinto.

FX FY identificano la direzione dritta.
LX LY identificano la direzione laterale: a sinistra, se sommate; a destra, se sottratte.
P1 e Q1 posizione dell'uscita.
LV livello di gioco.

3D color maze versione per C16 e Plus/4

```

20 DIMA$(27),P(8,2) :rem 190
50 GOTO7000 :rem 101
100 GOSUB1000 :rem 210
110 CHAR,12,22,"{RVS}TEMPO ":PRINTLEFTS(T
IS,2) :MID$(TI$,3,2) :MID$(TI$,5,2)
"OFF" :rem 202
130 GETB$:IFB$<>"{SIN}"ANDB$<>"{SU}"ANDB$
<>"{DES}"THEN110 :rem 193
140 IFB$="{SU}"THEN200 :rem 93
150 IFB$="{DES}"THENEND=D+1 :rem 121
160 IFB$="{SIN}"THENEND=D-1 :rem 252
170 IFD>7THENEND=4 :rem 196
180 IFD<2THENEND=5 :rem 191
190 GOTO100 :rem 100
200 IFASC(MID$(A$(U+P(D,1)),V+P(D,2)))=65
THEN110 :rem 105
210 U=U+P(D,1):V=V+P(D,2):GOTO100 :rem 124
1000 PRINT"{CLR}":FX=P(D,1):FY=P(D,2):LX=
P(D-1,1):LY=P(D-1,2) :rem 77
1010 X=U:Y=V:GOSUB9000 :rem 85
1020 ONHHGOSUB2000,2130 :rem 36
1030 GOSUB9100 :rem 14
1040 ONHHGOSUB2010,2140 :rem 40
1050 X=X+FX:Y=Y+FY :rem 111
1060 IFMID$(A$(X),Y,1)="A"THENHH=1-(X=P1A
NDY=Q1):ONHHGOTO3000,3030 :rem 88
1070 GOSUB9000:ONHHGOTO2020,2150 :rem 176
1080 GOSUB9100:ONHHGOTO2030,2160 :rem 180
1090 X=X+FX:Y=Y+FY :rem 115
1100 IFMID$(A$(X),Y,1)="A"THENHH=1-(X=P1A
NDY=Q1):ONHHGOTO3060,3090 :rem 95
1110 GOSUB9000:ONHHGOTO2040,2170 :rem 175
1120 GOSUB9100:ONHHGOTO2050,2180 :rem 179
1130 X=X+FX:Y=Y+FY :rem 110
1140 IFMID$(A$(X),Y,1)="A"THENHH=1-(X=P1A
NDY=Q1):ONHHGOTO3120,3150 :rem 93
1150 GOSUB9000:ONHHGOTO2060,2190 :rem 183
1160 GOSUB9100:ONHHGOTO2070,2200 :rem 178
1170 X=X+FX:Y=Y+FY :rem 114
1180 IFMID$(A$(X),Y,1)="A"THENHH=1-(X=P1A
NDY=Q1):ONHHGOTO3180,3210 :rem 100
1190 GOSUB9000:ONHHGOTO2080,2210 :rem 182
1200 GOSUB9100:ONHHGOTO2090,2220 :rem 177
1210 X=X+FX:Y=Y+FY :rem 109
1220 IFMID$(A$(X),Y,1)="A"THENHH=1-(X=P1A
NDY=Q1):ONHHGOTO3240,3270 :rem 98
1230 GOSUB9000:ONHHGOTO2100,2230 :rem 172
1240 GOSUB9100:ONHHGOTO2110,2240 :rem 176
1250 HH=ASC(MID$(A$(X+FX),Y+FY))-64 :rem 201
1260 ONHHGOSUB2250,2120 :rem 48
1410 CHAR,12,22,"{RVS}TEMPO ":PRINTLEFTS(
TI$,2) :MID$(TI$,3,2) :MID$(TI$,5,
2) "OFF" :rem 254
1420 IFLV=1ANDVAL(TI$)>=300THEN4000 :rem 47
1430 IFVAL(TI$)>=500THEN4000 :rem 79
1440 IFU=P1-1ANDV=Q1THEN4500 :rem 169

```

```

1450 IFLV=1ANDRND(0)>.5THEN1480 :rem 62
1460 CHAR,7,20,"{RED}DISTANZA DALL'USCITA
":PRINTINT(ABS(U-P1)+ABS(V-Q1))"
{BLK}" :rem 41
1480 RETURN :rem 173
2000 CHAR,9,0,"M":CHAR,9,18,"N":RETURN :rem 136
2010 CHAR,28,0,"N":CHAR,28,18,"M":RETURN :rem 235
2020 CHAR,10,1,"M":CHAR,11,2,"M":CHAR,10,
17,"N":CHAR,11,16,"N":RETURN :rem 18
2030 CHAR,27,1,"N":CHAR,26,2,"N":CHAR,27,
17,"M":CHAR,26,16,"M":RETURN :rem 47
2040 CHAR,12,3,"M":CHAR,13,4,"M":CHAR,12,
15,"N":CHAR,13,14,"N":RETURN :rem 28
2050 CHAR,25,3,"N":CHAR,24,4,"N":CHAR,25,
15,"M":CHAR,24,14,"M":RETURN :rem 41
2060 CHAR,14,5,"M":CHAR,15,6,"M":CHAR,14,
13,"N":CHAR,15,12,"N":RETURN :rem 38
2070 CHAR,23,5,"N":CHAR,22,6,"N":CHAR,22,
12,"M":CHAR,23,13,"M":RETURN :rem 35
2080 CHAR,16,7,"M":CHAR,16,11,"N":RETURN :rem 236
2090 CHAR,21,7,"N":CHAR,21,11,"M":RETURN :rem 229
2100 CHAR,17,8,"M":CHAR,17,10,"N":RETURN :rem 231
2110 CHAR,20,8,"N":CHAR,20,10,"M":RETURN :rem 220
2120 CHAR,17,9,"{<N>}{GRN}{RVS}
{ 2 SPAZI}{OFF}{BLK}{<H>":RETURN :rem 22
2130 CHAR,9,0,"{<P>":CHAR,9,18,"{<Y>":R
ETURN :rem 87
2140 CHAR,28,0,"{<P>":CHAR,28,18,"{<Y>":
RETURN :rem 186
2150 CHAR,9,1,"{<N>":CHAR,9,2,"{<N>}
{< 2 P>":FORJJ=3TO15:CHAR,9,JJ,"
{<N>}{RVS}{PUR}{ 2 SPAZI}{OFF}{BLK}"
:NEXT :rem 99
2155 CHAR,9,16,"{<N>}{< 2 Y>":CHAR,9,17,
"{<N>":RETURN :rem 240
2160 CHAR,28,1,"{<H>":CHAR,26,2,"
{< 2 P>}{<H>":FORJJ=3TO15:CHAR,26,J
J,"{RVS}{PUR}{ 2 SPAZI}{BLK}{OFF}
{<H>":NEXT :rem 17
2165 CHAR,26,16,"{< 2 Y>}{<H>":CHAR,28,1
7,"{<H>":RETURN :rem 101
2170 CHAR,11,3,"{<N>":CHAR,11,4,"{<N>}
{< 2 P>":FORJJ=5TO13:CHAR,11,JJ,"
{<N>}{RVS}{CYN}{ 2 SPAZI}{OFF}{BLK}"
:NEXT :rem 231
2175 CHAR,11,14,"{<N>}{< 2 Y>":CHAR,11,1
5,"{<N>":RETURN :rem 64
2180 CHAR,26,3,"{<H>":CHAR,24,4,"
{< 2 P>}{<H>":FORJJ=5TO13:CHAR,24,J
J,"{RVS}{CYN}{ 2 SPAZI}{OFF}{BLK}
{<H>":NEXT :rem 20
2185 CHAR,24,14,"{< 2 Y>}{<H>":CHAR,26,1
5,"{<H>":RETURN :rem 95
2190 CHAR,13,5,"{<N>":CHAR,13,6,"{<N>}
{< 2 P>":FORJJ=7TO11:CHAR,13,JJ,"
{<N>}{RVS}{<2>}{ 2 SPAZI}{OFF}{BLK}"
:NEXT :rem 233
2195 CHAR,13,12,"{<N>}{< 2 Y>":CHAR,13,1
3,"{<N>":RETURN :rem 66

```




```

2200 CHAR,24,5,"[<H>]":CHAR,22,6,"
[< 2 P>][<H>]":FORJJ=7TO11:CHAR,22,J
J,"[RVS][<2>]{ 2 SPAZI}{OFF}{BLK}
[<H>]":NEXT:rem 1
2205 CHAR,22,12,"[< 2 Y>][<H>]":CHAR,24,1
3,"[<H>]":RETURN:rem 80
2210 CHAR,15,7,"[<N>][<P>]":FORJJ=8TO10:C
HAR,15,JJ,"[<N>][RVS]{BLU}{OFF}
{BLK}":NEXT:CHAR,15,11,"[<N>][<Y>]":
RETURN:rem 194
2220 CHAR,21,7,"[<P>][<H>]":FORJJ=8TO10:C
HAR,21,JJ,"[RVS]{BLU}{OFF}{BLK}
[<H>]":NEXT:CHAR,21,11,"[<Y>][<H>]":
RETURN:rem 216
2230 CHAR,16,8,"[<N>][<P>]":CHAR,16,9,"
[<N>][RVS]{CYN}{OFF}{BLK}":CHAR,16,
10,"[<N>][<Y>]":RETURN:rem 69
2240 CHAR,20,8,"[<P>][<H>]":CHAR,20,9,"
{RVS}{CYN}{OFF}{BLK}[<H>]":CHAR,20,
10,"[<Y>][<H>]":RETURN:rem 85
2250 CHAR,18,8,"[< 2 P>]":CHAR,18,9,"
{RVS}{CYN}{ 2 SPAZI}{OFF}{BLK}":CHAR
,18,10,"[< 2 Y>]":RETURN:rem 181
3000 CHAR,10,0,"[< 18 P>]":FORJJ=1TO17
:rem 213
3010 CHAR,10,JJ,"[RVS][<1>]{ 18 SPAZI}":N
EXT:rem 171
3020 CHAR,10,18,"[BLK]{OFF}[< 18 Y>]":GOT
O1410:rem 174
3030 FORJJ=1TO17:CHAR,10,JJ,"[RVS]{RED}
{ 18 Z}":NEXT:PRINT"[BLK]":GOTO1410
:rem 158
3060 CHAR,12,2,"[< 14 P>]":FORJJ=3TO15:CH
AR,12,JJ,"[RVS]{PUR}{ 14 SPAZI}":NEX
T:rem 66
3070 PRINT"[OFF]{BLK}":CHAR,12,16,"
[< 14 Y>]":GOTO1410:rem 226
3090 FORJJ=3TO15:CHAR,12,JJ,"[RVS]{RED}
{ 14 Z}":NEXT:PRINT"[OFF]{BLK}":GOTO
1410:rem 208
3120 CHAR,14,4,"[< 10 P>]":FORJJ=5TO13:CH
AR,14,JJ,"[RVS]{CYN}{ 10 SPAZI}":NEX
T:rem 140
3130 PRINT"[OFF]{BLK}":CHAR,14,14,"
[< 10 Y>]":GOTO1410:rem 3
3150 FORJJ=5TO13:CHAR,14,JJ,"[RVS]{RED}
{ 10 Z}":NEXT:PRINT"[OFF]{BLK}":GOTO
1410:rem 103
3180 CHAR,16,6,"[< 6 P>]":FORJJ=7TO11:CHA
R,16,JJ,"[RVS][<2>]{ 6 SPAZI}{OFF}":
NEXT:rem 100
3190 PRINT"[OFF]{BLK}":CHAR,16,12,"
[< 6 Y>]":GOTO1410:rem 45
3210 FORJJ=7TO11:CHAR,16,JJ,"[RVS]{RED}
{ 6 Z}":NEXT:PRINT"[OFF]{BLK}":GOTO1
410:rem 254
3240 CHAR,17,7,"[< 4 P>]":FORJJ=8TO10:CHA
R,17,JJ,"[RVS]{BLU}{ 4 SPAZI}":NEXT
:rem 254

```

```

3250 PRINT"[OFF]{BLK}":CHAR,17,11,"
[< 4 Y>]":GOTO1410:rem 188
3270 FORJJ=8TO10:CHAR,17,JJ,"[RVS]{RED}
{ 4 Z}":NEXT:PRINT"[OFF]{BLK}":GOTO1
410:rem 81
4000 CHAR,12,23,"[RVS]{FLASH ON}TEMPO
{ 2 SPAZI}SCADUTO":rem 158
4010 VOL8:SOUND1,200,160:FORI=7TO2STEP-1
:rem 176
4020 VOLI:FORJ=1TO50:NEXT:NEXT:FORK=1TO20
00:NEXT:rem 139
4030 CHAR,9,23,"[RVS]{BLK}GIOCHI ANCORA?
(S/N)":PRINT:rem 123
4040 GETZ$:IFZ$="S"THEN7500:rem 60
4050 IFZ$<"S"THEN4040:rem 217
4060 PRINT"[CLR]":END:rem 63
4500 CHAR,6,20,"[<7>]BRAVO HAI TROVATO L'
USCITA!":FORK=1TO10:rem 151
4510 VOL8:FORI=4TO8:FORJ=300TO500STEP100:
SOUND2,J,2:NEXT:rem 43
4520 COLORO,I:COLOR4,I+8:NEXTI,K:rem 12
4530 COLOR4,16,6:COLOR0,2:rem 187
4540 GOTO4030:rem 205
7000 COLOR4,16,6:COLOR0,2:rem 182
7010 PRINT"[CLR]":CHAR,13,5,"[<2>]3D COLO
R MAZE":rem 157
7020 CHAR,7,8,"[<7>]VUOI LE ISTRUZIONI (S
/N)?" :rem 12
7030 GETZ$:IFZ$="N"THEN7500:rem 57
7040 IFZ$<"S"THEN7020:rem 225
7050 PRINT"[HOME]{ 8 GIU' }SEI STATO IMPRI
GIONATO IN UN LABIRINTO E":rem 89
7060 PRINT"DEVI USCIRNE PRIMA CHE IL TEMP
O A DISPO-":rem 47
7070 PRINT"SIONE FINISCA":rem 189
7080 PRINT"PER MUOVERE USA I CURSORI (DES
TRO - ALTO)":rem 89
7090 PRINT"E SINISTRO) CHE INDICANO LE RE
LATIVE DI-":rem 70
7100 PRINT"REZIONI":rem 185
7110 PRINT"[GIU']{ 6 DES}{RVS}PREMI UN TA
STO PER CONTINUARE":rem 245
7120 GETZ$:IFZ$="N"THEN7120:rem 233
7500 PRINT"[CLR]{BLK}{ 2 GIU' }{ 11 DES}LI
VELLO DI GIOCO?":rem 117
7510 CHAR,12,5,"(P)RINCIPIANTE":CHAR,12,7
,"(E)SPERTO":rem 79
7520 GETZ$:IFZ$="E"THENLV=1:GOTO7550
:rem 188
7530 IFZ$<"P"THEN7520:rem 231
7540 LV=0:rem 223
7550 CHAR,2,10,"[RED]ATTENDI MENTRE PREPA
RO IL LABIRINTO[BLK]":rem 28
8000 Z$="21120110":FORJJ=1TO4:rem 73
8010 P(JJ,1)=VAL(MID$(Z$,(JJ*2-1),1))-1
:rem 31
8020 P(JJ,2)=VAL(MID$(Z$,(JJ*2),1))-1
:rem 195
8030 P(JJ+4,1)=P(JJ,1):P(JJ+4,2)=P(JJ,2)
:rem 135
8040 NEXT:Z$="ABBBBBBBBBBBBBBBBBBBBBBBA
":rem 56
8050 FORJJ=1TO27:A$(JJ)="AAAAAAAAAAAAAAAA
AAAAAAAAA":NEXT:rem 27
8060 FORJJ=2TO22STEP4:A$(JJ)=Z$:rem 110
8070 MID$(A$(JJ+1),2,3)="B":MID$(A$(JJ+2)
,2,3)="B":rem 244
8080 MID$(A$(JJ+3),2,3)="B":MID$(A$(JJ+1)
,26,27)="B":rem 98
8090 MID$(A$(JJ+2),26,27)="B":MID$(A$(JJ+
3),26,27)="B":rem 208
8100 FORKK=2TO22STEP4:HH=INT(RND(0)*1.6):

```

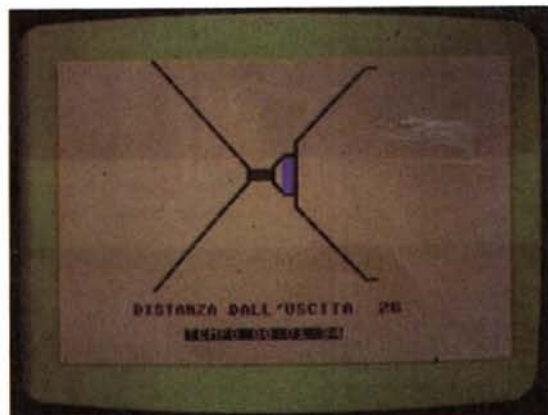



**3D color maze
per C16**

```

ONHHGOTO8110,8120                                :rem 244
8110 MIDS(A$(JJ+1),KK,KK+1)="B"                   :rem 98
8120 MIDS(A$(JJ+3),KK,KK+1)="B"                   :rem 101
8130 IFRND(0)>.5AND(ASC(MIDS(A$(JJ+1),KK))
)=66ORASC(MIDS(A$(JJ+3),KK))=66) THEN
8500                                              :rem 52
8140 NEXTKK,JJ:A$(26)=Z$                          :rem 117
8150 P1=14+4*INT(RND(0)*3):Q1=6+4*INT(RND
(0)*5)                                             :rem 48
8160 MIDS(A$(P1+1),Q1,Q1+1)="B":MIDS(A$(P
1+2),Q1,Q1+1)="A"                               :rem 195
8170 MIDS(A$(P1+3),Q1,Q1+1)="A":MIDS(A$(P
1-1),Q1,Q1+1)="B"                               :rem 199
8180 MIDS(A$(P1-2),Q1,Q1+1)="B":MIDS(A$(P
1-3),Q1,Q1+1)="B"                               :rem 204
8190 MIDS(A$(P1-5),Q1,Q1+1)="B":MIDS(A$(P
1),Q1-1,Q1)="A"                                  :rem 113
8200 MIDS(A$(P1),Q1+1,Q1+2)="A":MIDS(A$(P
1-6),Q1,Q1+1)="B"                               :rem 197
8210 MIDS(A$(P1-7),Q1,Q1+1)="B":P1=P1+2
                                                :rem 6
8220 U=2+24*INT(RND(0)*2):V=2:D=5-(U=26)
                                                :rem 175
8230 Z=4+4*INT(RND(0)*6):FORJJ=1TO3
                                                :rem 27
8240 MIDS(A$(6+4*INT(RND(0)*5)),Z,Z+1)="A"
":NEXT                                           :rem 53
8250 TI$="0000000"                              :rem 49
8260 GOTO100                                       :rem 154
8500 MIDS(A$(JJ+2),KK,KK+1)="B":GOTO8140
                                                :rem 166
9000 HH=ASC(MIDS(A$(X+LX),Y+LY))-64:RETUR
N                                                  :rem 240
9100 HH=ASC(MIDS(A$(X-LX),Y-LY))-64:RETUR
N                                                  :rem 245

```



CON
CASSETTA



Cod. 553D Pag. 164 Lire. 24.000

Ogni argomento viene spiegato e accompagnato da numerosi esempi commentati. Nel libro sono listati moltissimi programmi; essi sono registrati sulla cassetta allegata. Inoltre la cassetta contiene anche altri programmi, ai quali si fa riferimento nel testo, ma senza listarli.

Cod. 4098 Pag. 270 — Lira. 34.000



ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON - Divisione Libri - Via Rosellini, 12 - 20124 Milano

CEDOLA DI COMMISSIONE LIBRARIA**VOGLIATE SPEDIRMI**

n° copie	codice	Prezzo unitario	Prezzo totale
Totale			

Total

☐ **Pagherò contrassegno al postino il prezzo indicato più L. 3.000 per contributo fissa spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca

☐ **Allego fotocopia del versamento su c/c n. 11666203 a voi intestato**

2

☐ **Allego fotocopia di versamento su vaglia postale a voi intestato**

Norms

Concours

View

GMA

Procy

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

**ORDINE
MINIMO
L. 50.000**



Apple Hunt

di J. Arlet

trad. ed adatt.
di M. Cristofa Grizzi

La velocità è importante, ma sarà la buona strategia delle mosse ad assicurare il successo in questo simpatico gioco in versioni per VIC 20 inespanso, C16-Plus/4 e C64. È richiesto l'uso di un joystick.

Rappresentati da una simpatica faccina, vi trovate in una foresta ove sono sparse qua e là alcune mele. Tremendamente affamati, il vostro scopo è mangiarne la maggior quantità possibile. Dovrete però cercare di evitare di sbattere contro gli alberi e di essere a vostra volta divorati da un mostro in movimento, che si ciba di ogni cosa: alberi, mele e... uomini. Oltre a ciò, il tempo a vostra disposizione è limitato: lunghe pause per studiare la situazione non sono quindi raccomandabili. L'unica via per avere successo in questo gioco consiste nel pensare prima di muovere, ma senza pensare troppo a lungo...

La vostra sola arma, in questo gioco, è una bomba che distrugge tutto ciò che si trova nelle otto locazioni di schermo immediatamente adiacenti alla vostra posizione. La bomba viene lanciata con la pressione del pulsante di fuoco, ed è utile per aprire la strada dagli alberi nel tentativo di raggiungere qualche mela in posizione difficile, oppure per mettere fuori combattimento il mostro. Casualmente, può apparire un rubino sullo schermo. Dovrete cercare di raccoglierlo prima che lo faccia il mostro, e vi renderà un bel po' di punti.

Una volta attivato il programma vi verrà domandato se desiderate leggere le istruzioni di gioco. Usando un joystick (porta 2 nel C64, C16 e Plus/4), muovete la faccina verso una mela o verso il rubino (se presente) ed usate il pulsante di fuoco per lanciare una bomba. Uscendo dal bordo dello schermo, rientrerete automaticamente dalla parte opposta.

Il gioco inizia nel livello 1 con due uomi-



ni e due bombe a disposizione. Per completare ciascun livello dovete mangiare cinque mele, oppure raccogliere il rubino, oppure uccidere il mostro con la bomba; tutto ciò entro il limite di tempo massimo. Ogni livello successivo è caratterizzato da un numero superiore di alberi, il che rende il movimento sempre più difficile. Poco prima del termine del tempo a disposizione il bordo dello schermo diventa rosso, per indicarvi che rimangono soltanto pochi secondi a disposizione. L'uso di una bomba ripor-

ta all'inizio il tempo disponibile in quel dato livello di gioco.

I punti sono distribuiti come segue: da 50 a 500 per le mele, 1000 se uccidete il mostro e 3000 se raccogliete il rubino. Inoltre, punti di bonus vengono regalati in funzione della velocità con cui completate un livello di gioco.

Infine si guadagnano un omino ed una bomba extra dopo i livelli 5, 10, 15, 20 e 25. Buon divertimento!

Apple Hunt versione per VIC 20

Programma 1

```

10 POKE55,152:POKE56,28:CLR          :rem 67
20 FORI=1TO95:READA:X=X+A:NEXT        :rem 180
25 IFX<>12828THENPRINT"ERRORE NELLE ISTRU
   ZIONI DATA!":STOP                :rem 231
30 RESTORE:FORI=828TO866:READA:POKEI,A:NE
   XT                                :rem 70
40 DATA 173,5,144,41,3,10,10        :rem 73
50 DATA 105,16,133,254,169,128      :rem 197
60 DATA 133,252,169,0,133,251,133   :rem 78
70 DATA 253,168,162,2,177,251,145,253 :rem 36
80 DATA 200,208,249,230,252,230,254 :rem 179
90 DATA 202,208,242,96               :rem 60
100 POKE36878,15:DIMN(16):FORI=1TO16:READ
    N(I):NEXT                        :rem 21
110 DATA 135,163,175,195,207,209,215,225,
    231,232,235,240,237,235,237,240 :rem 217
120 POKE36879,25:PRINT"{CLR}"CHR$(142)CHR
    $(8)SPC(4)"{ 6 GIU'}{BLK}BENVENUTI IN
    "                                :rem 99
130 PRINTSPC(5)"{ 2 GIU'}{BLK}*{RED}A
    {CYN}P{PUR}P{GRN}L{BLU}E{YEL}H{BLK}U
    {RED}N{CYN}T{PUR}*":GOSUB600:rem 241
140 PRINT"{BLU}{ 4 GIU'}{RVS}VUOI LE ISTRU
    ZIONI?":PRINT"(S O N)"          :rem 90
150 GETY$:IFY$=""OR(Y$<>"S"ANDY$<>"N")THE
    N150                              :rem 91
160 Y=(Y$="S"):PRINT"{CLR}":POKE36869,255
    :SYS828:GOSUB500:IFNOT Y THEN440 :rem 178
170 PRINT"{CLR}":POKE36879,251      :rem 58
180 PRINTSPC(5)"{RVS}{PUR}TU SEI :{OFF}
    {BLK}S":PRINTSPC(5)"{GIU'}{RVS}{PUR}M
    OSTRO:{OFF}{BLU}!":PRINTSPC(5)"
    {GIU'}{RVS}{PUR}MELA:{OFF}{RED}V" :rem 118
190 PRINTSPC(5)"{GIU'}{RVS}{PUR}ALBERO:
    {GRN}{OFF}U":PRINTSPC(5)"{GIU'}{RVS}
    {PUR}RUBINO:Z"                  :rem 163
200 PRINT"{ 2 GIU'}{RVS}{BLK}PUNTEGGI:"P
    RINT"{GIU'}{RED}V{RVS}:50-500 PUNTI":
    PRINT"{GIU'}{BLU}!{RVS}:1000 PUNTI" :rem 78
210 PRINT"{GIU'}{PUR}{RVS}Z:3000 PUNTI" :rem 138
220 GOSUB640                        :rem 174
230 PRINT"{CLR}{GIU'}{BLK}S {RVS}{BLU}DEV
    E PRENDERE 5 {RED}{OFF}V{RVS}{BLU}
    { 3 SPAZI}PRIMA DELLO SCADERE
    { 3 SPAZI}DEL TEMPO"            :rem 59
240 PRINT"{BLK}{RVS}IL {BLU}{OFF}!{BLK}
    {RVS}MANGIA SIA IL {RED}{OFF}V{RVS}
    {BLK}{ 2 SPAZI}CHE IL {GRN}{OFF} U
    {BLK}"                          :rem 209
250 PRINT"{GIU'}{RVS}{BLK}AD OGNI LIVELLO
    ,{ 6 SPAZI}AUMENTANO GLI OFFI{GRN}U
    {BLK}"                          :rem 7
260 PRINT"{GIU'}{RVS}{BLU}USA IL JOYSTICK
    PER{ 3 SPAZI}MUOVERE {OFF}{BLK}S
    {RVS}{BLU} NELLE 4{ 5 SPAZI}DIREZIONI
    "                                :rem 154
270 GOSUB640                        :rem 179
280 PRINT"{CLR}{ 2 GIU'}{ 3 DES}{BLK}S
    {RVS}{BLU} PERDERA' SE"         :rem 77
290 PRINTSPC(3)"{GIU'}{RVS}TOCCA UN {GRN}
    {OFF}U{RVS}{BLU} O SE"         :rem 35

```

```

300 PRINTSPC(3)"{GIU'}{RVS}{BLU}E'TOCCATO
    DA {OFF} !{BLK}"               :rem 81
310 PRINT"{ 2 GIU'}{RVS}{RED}IL GIOCO FIN
    ISCE SE {BLK}{OFF}S{RVS}":PRINT"{RVS}
    {RED}CI IMPIEGA TROPPO!"       :rem 161
320 GOSUB640                        :rem 175
330 PRINT"{CLR}{ 2 GIU'}{BLU}{RVS}EXTRA
    {OFF}{BLK}S{RVS}{BLU} E BOMBE EXTRA,A
    I LIVELLI 5,10,15,20 E 25"    :rem 250
340 PRINT"{ 2 GIU'}{BLU}{RVS}SE {OFF}
    {BLK}S{RVS}{BLU} STA ESAURENDO IL TEM
    PO, LO SCHERMO{ 5 SPAZI}DIVENTA {RED}
    ROSSO"                          :rem 54
350 PRINT"{ 2 GIU'}{BLU}{RVS}PIU' VELOCEM
    ENTE {OFF}{BLK}S{RVS}{BLU}{ 4 SPAZI}
    COMPLETA UN LIVELLO,{ 2 SPAZI}PIU' PU
    NTI OTTIENE"                   :rem 166
360 GOSUB640                        :rem 179
370 PRINT"{CLR}{ 2 GIU'}{RVS}{BLU}PREMEND
    O FUOCO SI{ 5 SPAZI}GETTA UNA BOMBA" :rem 125
380 PRINT"{GIU'}{BLK}S {RVS} PUO' GETTARE
    { 3 SPAZI}UNA SOLA BOMBA ALLA VOLTA" :rem 173
390 PRINT"{GIU'}{RVS}{RED}UNA BOMBA DISTR
    UGGE {OFF}{GRN}U{RED}{RVS} E REINIZIA
    LIZZA IL{ 4 SPAZI}TEMPO"       :rem 22
400 PRINT"{GIU'}{RVS}{BLK}LA BOMBA SPOSTA
    "                                :rem 27
410 PRINT" {OFF}{BLU}!{BLK}{RVS} IN
    { 2 SPAZI}UN ALTRO LUOGO, O FA
    { 2 SPAZI}SCOMPARE {OFF}{BLU}!{BLK}
    "                                :rem 117
420 PRINT"{ 2 GIU'}{DES}{RVS}PREMI UN TAS
    TO"                             :rem 99
430 GETAS:IFAS=""THEN430           :rem 81
440 PRINT"{CLR}":POKE36879,25:POKE36869,2
    40                              :rem 68
450 PRINT                          :rem 38
460 PRINT"{ 2 GIU'}ATTENDERE:CARICO LA SE
    CONDA PARTE"                   :rem 26
470 SS=CHR$(131)                   :rem 54
490 FORI=1TOLEN(SS):POKE630+I,ASC(MID$(SS
    ,I)):NEXT:POKE198,I:END        :rem 148
500 FORC=7320TO7327:READA:POKEC,A:NEXT
    :rem 103
510 DATA 60,66,165,129,165,153,66,60 :rem 245
520 FORC=7432TO7439:READA:POKEC,A:NEXT :rem 113
530 DATA 195,34,20,156,126,29,8,119 :rem 193
540 FORC=7328TO7335:READA:POKEC,A:NEXT :rem 114
550 DATA 60,66,165,129,153,165,66,60 :rem 249
560 FORC=7336TO7343:READA:POKEC,A:NEXT :rem 114
570 DATA 0,28,62,127,127,62,8,8    :rem 248
580 FORC=7344TO7351:READA:POKEC,A:NEXT:RE
    TURN                            :rem 140
590 DATA 6,8,60,126,126,126,60,24 :rem 87
600 POKE36878,15:FORO=0TO8:FORL=1+OTO8+O:
    N=N(L)                          :rem 121
610 POKE36876,N:FORO=1TO50:NEXT:NEXT:
    NEXT:FORV=15TOOSTEP-1          :rem 43
615 POKE36878,V:FORO=1TO300:NEXT:rem 239
620 POKE36878,0:POKE36876,0        :rem 6
630 RETURN                          :rem 121
640 PRINT"{ 2 GIU'}{BLK}{RVS}{ 5 DES}PREM
    I UN TASTO"                    :rem 107

```



```

650 GETAS:IFAS=""THEN650          :rem 89
660 RETURN                          :rem 124

```

Programma 2

```

2 S1=36874:S2=S1+1:S3=S2+1:V=S3+2:TP=3687
9                                     :rem 149
4 GOSUB110:B#=0:SO=0:TX=0:X=RND(0)
                                     :rem 36
6 PRINT"{CLR}":POKE36869,255:SC=7680:Q=30
720:ZZ=25:W=1:ME=2:POKEV,15:ZA=2:POKE37
154,127                             :rem 21
8 POKETP,251:NO=127:FORL=1TOZZ:X=INT(RND(
1)*506)                             :rem 229

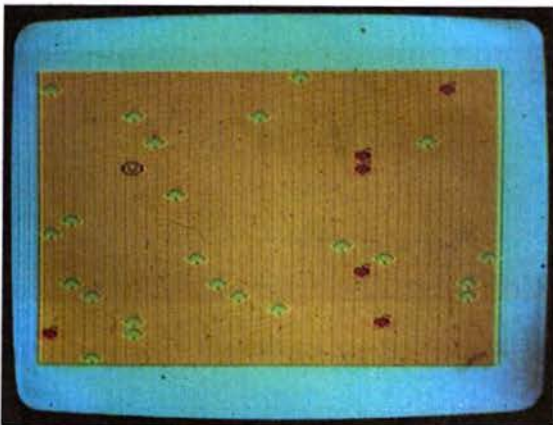
10 NO=NO+1:POKES2,NO:IFNO=>254THENNO=127
                                     :rem 10
12 POKESC+Q+X,5:POKESC+X,21:POKES2,0:NEXT
:ZZ=ZZ+15:TR=INT(RND(1)*4)+6 :rem 74
14 FORP=1TOTR:POKES3,240:TA=INT(RND(1)*50
6)+SC:POKETA,22:POKETA+Q,2 :rem 158
16 IFINT(RND(1)*100)+1<=3THENPOKETA,218:P
OKETA+Q,4 :rem 239

18 POKES3,0:NEXT:SC=INT(RND(1)*506)+SC:PO
KESX,33:POKESX+Q,6 :rem 213
20 PO=INT(RND(1)*506)+SC:IFPO=SCTHEN20
:rem 71
22 POKEPO,19:POKEPO+Q,0:FORL=1TO3:POKEPO,
19:FORP=1TO100:POKES2,200:NEXT
:rem 47
24 POKEPO,32:FORP=1TO100:POKES2,0:NEXT:NE
XT:POKEPO,19 :rem 178
26 TI$="000000" :rem 202

28 IFTI=>786THENPOKETP,154:IFTI=>966THEN6
2 :rem 155
30 POKES3,0:J=(PEEK(37137)AND28)OR(PEEK(3
7152)AND128) :rem 22
32 ONABS((J-100)/4)-7GOTO50,50,34,36,38,
124,,,40,50,50 :rem 92
34 DR=-1:NO=165:GOTO42 :rem 33
36 DR=22:NO=189:GOTO42 :rem 47

38 DR=-22:NO=210:GOTO42 :rem 79
40 DR=1:NO=225 :rem 21
42 POKES2,NO:PO=PO+DR:IFPO>8187THENPO=PO-
506:POKEPO-DR+506,32 :rem 107
44 IFPO<7680THENPO=PO+506:POKEPO-DR-506,3
2 :rem 59

```



```

46 IFPEEK(PO)<>32THEN64 :rem 134
48 POKEPO-DR,32:POKEPO,19:POKEPO+Q,0:POKE
S2,0 :rem 86
50 POKES1,128:POKESX,32:X=INT(RND(1)*8)+1
:rem 179
52 JJ=(X=2)*22-(X=1)*22-(X=3)+(X=4)+(X=5)
*23-(X=6)*23+(X=7)*21-(X=8)*21
:rem 149
54 SX=SX+JJ:IFSX>7680THENSX=SX+506:POKESX
-JJ,32 :rem 46
56 IFSX>8187THENSX=SX-506:POKESX-JJ,32
:rem 171
58 IFPEEK(SX)=19THENPOKESX-JJ,33:POKESX+Q
-JJ,6:POKEPO,20:POKEPO+Q,0:GOTO78
:rem 43
60 POKESX,33:POKESX+Q,6:POKES1,0:GOTO28
:rem 234
62 PRINT"{ 2 GIU' }{RVS}{BLK}TROPPO TARDI"
:POKEPO,20:POKEPO+Q,0:GOTO82 :rem 168
64 IFPEEK(PO)=21ORPEEK(PO)=33THENPOKEPO-D
R,20:POKEPO+Q-DR,0:GOTO78 :rem 218
66 IF PEEK(PO)<>22THEN72 :rem 134
68 POKEPO-DR,32:POKEPO,19:POKEPO+Q,0:SO=S
O+INT(RND(1)*10+1)*50:POKES3,230+B#:TX
=TX+1 :rem 90
70 IFTX>4THEN94 :rem 176
72 IF PEEK(PO)<>218THENB#=B#+2:GOTO48
:rem 152
74 POKEPO-DR,32:POKEPO,19:POKEPO+Q,0
:rem 11
76 PRINT"{HOME}{RVS}{ 2 DES }{BLK}3000 PUN
TI!":SO=SO+3000:A#=1:POKES3,250:GOTO9
4 :rem 27
78 POKES1,0:POKES3,0:FORP=0TO53:POKES2,18
0-P:FORL=1TO20:NEXT:NEXT :rem 180
80 ME=ME-1:IFME=>1THENZZ=ZZ-15:PRINT"
{CLR}":GOTO8 :rem 222
82 PRINT"{HOME}{RVS}{ 4 GIU' }{BLK}":FORP=
1TO10:PRINT"{RVS}{BLK}GAME OVER":POKES
1,128:FORL=1TO100:NEXT :rem 86
84 PRINT"{SU}{ 8 SPAZI }{SU}":POKES1,0:FOR
L=1TO100:NEXT:NEXT:IFSO>HSTHENHS=SO
:rem 150
86 POKE36869,240:POKE198,0:PRINT"{CLR}
{BLK}{ 2 GIU' }PUNTI="SO:PRINT"{GIU'}LI
V."W:PRINT"{GIU'}{BLK}ANCORA?(S O N)"
:rem 1
88 GETC$:IFC$="N"THENPOKE37139,128:POKE37
154,255:END :rem 40
90 IFC$="S"THEN4 :rem 150
92 GOTO88 :rem 20
94 BO=INT(966-TI)*3:IFPEEK(TP)=154THENBO=
0 :rem 42
96 PRINT"{HOME}{ 3 GIU' }{RVS}{BLK}LIV."W
;"FINITO":TX=0:IFW=50RW=10ORW=150RW=20
ORW=25THEN118 :rem 45
98 W=W+1:PRINT"{RVS}{GIU'}BONUS="BO:SO=SO
+BO:PRINT"{RVS}{GIU'}PUNTI="SO
:rem 79
100 PRINT"{RVS}{GIU'}VITE":ME=POKES1,0:PO
KES2,0:POKES3,0:POKES3+1,0:PRINT"
{RVS}{GIU'}BOMBE"ZA:B#=0 :rem 75
102 READN,D:IFN=-1THENPOKES2+A#,0:A#=0:
GOTO108 :rem 106
104 POKES2+A#,N:FORP=1TOD:NEXT:GOTO102
:rem 202
107 DATA 0,5,220,120,0,5,230,450,-1,-1
:rem 25
108 FORP=1TO2500:NEXT:PRINT"{CLR}":RESTOR
E:GOTO8 :rem 200

```



```

110 POKETP,30:PRINT{CLR}{ 3 GIU' }{RED}"S
PC(6)"APPLE HUNT{ 3 GIU' }":PRINT"
{ 3 GIU' }{ 3 DES }{BLK}MAX.PUNTEGGIO="
HS :rem 57
112 PRINT" 4 GIU' }{DES}{PUR}{RVS}PREMI U
N TASO" :rem 31
114 GETAS:IFAS="THEN114 :rem 79
116 RETURN :rem 120
118 POKES2,0:FORP=1TO10:PRINT{HOME}{RVS}
VITA E BOMBA EXTRA":POKES3,240:FORL=1
TO100:NEXT :rem 84
120 PRINT{SU}{ 16 SPAZI }":POKES3,0:FORL=
1TO100:NEXT:PRINT{ 2 GIU' }" :rem 110
122 ME=ME+1:ZA=ZA+1:GOTO98 :rem 63
124 B=PEEK(37137)AND32:IFB<>0THEN50 :rem 18
126 IFZA=<0THEN50 :rem 0
128 ZA=ZA-1:POKETP,203 :rem 58
130 POKEPO+1,192:POKEPO-1,192:POKEPO-22,2
21:POKEPO+22,221 :rem 102
132 POKEPO+21,206:POKEPO-21,206:POKEPO-23
,205:POKEPO+23,205 :rem 202
134 POKEPO+1+Q,0:POKEPO-1+Q,1:POKEPO-22+Q
,2:POKEPO+22+Q,3 :rem 190
136 POKEPO+21+Q,4:POKEPO-21+Q,5:POKEPO-23
+Q,6:POKEPO+23+Q,7 :rem 54
138 FORL=1TO100:POKES3+1,150:NEXT:POKES3+
1,0 :rem 158
140 IFPEEK(SX)<>33THENS0=SO+1000:POKES3,2
40:PRINT{HOME}{RVS}{BLK}1000 PUNTI"
:GOTO94 :rem 132
142 POKESX,32:SC=SC+INT(RND(1)*506)+1:POK
EPO+1,32:POKEPO-1,32:POKEPO-22,32:POK
EPO+22,32 :rem 230
144 POKEPO+21,32:POKEPO-21,32:POKEPO-23,3
2:POKEPO+23,32:POKETP,251:TIS="000000
":GOTO50 :rem 72

```

Apple Hunt versione per C16

```

100 TP=65305:S=-1024:VOL8 :rem 150
110 FORI=1TO88:READA:X=X+A:NEXT :rem 230
120 IFX<93099THENPRINT{CLR}{BLK}ERRORE
NELLE ISTR.DATA11":STOP :rem 176
130 RESTORE :rem 184
140 POKETP,12:POKETP-4,12:PRINT{CLR}
{WHT}{ 9 GIU' }"TAB(10)"BENVENUTI NELL
A" :rem 25
150 PRINTTAB(12){ 4 GIU' }{BLK}*{RED}APPL
E{BLU}HUNT{BLK}":FORI=1TO1000:NEXT:C
OLOR0,2:COLOR1,2:GOTO1270 :rem 23
160 GOSUB430:POKE65298,PEEK(65298)AND251
:rem 64
170 POKE65299,56:COLOR1,1 :rem 180
180 POKE52,55:POKE56,55 :rem 47
190 PRINT{BLK}{ 4 GIU' }"TAB(6)"{RVS}VUOI
LE ISTRUZIONI? (S/N)" :rem 71
200 GETKEY Z$:IFZ$="OR(Z$<>"S"ANDZ$<>"N"
)THEN200 :rem 64
210 IFZ$="N"THEN480 :rem 58
220 POKETP,1:POKETP-4,1 :rem 143
230 AS="{CLR}{BLK}{ 4 GIU' }":PRINTAS"TU S
EI %. SCOPO DEL GIOCO E' " :rem 160
240 PRINT"MANGIARE{ 3 SPAZI }TUTTE LE MELE
{RED}{BLK}" :rem 83
250 PRINT"NEL BOSCO. DEVI EVITAREGLI ALBE
RI{GRN}{BLK} E " :rem 136
260 PRINT"I MOSTRI{BLU}{BLK}. USA IL J
OY-STICK PER MUOVERE & ATTRAVERSO IL
BOSCO." :rem 109

```

LOAD

Apple Hunt

```

270 PRINT"PUOI MUOVERTI NELLE QUATTRO DIR
EZIONI." :rem 157
280 PRINT"% HA UN TEMPO LIMITATO PER MANG
IARE{ 5 SPAZI }TUTTE LE {RED}&{BLK}. L
O SCHERMO " :rem 221
290 PRINT"DIVENTA ROSSO SE % STA PER FINI
RE IL SUO TEMPO." :GOSUB1240 :rem 22
300 PRINTAS{BLU}{BLK} MANGERA' SIA
{GRN}{BLK} CHE {RED}&{BLK}." :rem 172
310 PRINT"SE % RAGGIUNGE {BLU}{BLK} O
{GRN}{BLK}, HAI PERSO." :rem 152
320 PRINT"PREMI FUOCO PER BUTTARE UNA BOM
BA. UNA{ 2 SPAZI }BOMBA PUO' DISTRUGGE
RE " :rem 176
330 PRINT{GRN}{BLK} E REINIZIALIZZAI T
EMPO. SE {BLU}{BLK} VIENE COLPITO DA
UNA BOM-BA, " :rem 57
340 PRINT"GUADAGNI 1000 PUNTI." :rem 39
350 PRINT{BLU}{BLK} VAGHERA' ATTORNO A
{RED}&{BLK} FINO A CHE % LO
{ 2 SPAZI }MANGI O {BLU}{BLK} LO RAGG
IUNGA." :rem 126
360 GOSUB1240 :rem 224
370 PRINTAS"A VOLTE UN {PUR}{BLK} APPARE
SULLO SCHERMO:{ 2 SPAZI }DEVICERCARE
DI " :rem 5
380 PRINT"PRENDERE {PUR}{BLK} PRIMA DI
{BLU}{BLK}. PUOI{ 2 SPAZI }GUADAGNARE
3000 PUNTI SE PRENDI " :rem 8
390 PRINT{PUR}{BLK}. PER{ 2 SPAZI }OGNI
{RED}&{BLK} PUOI OTTENERE " :rem 174
400 PRINT"TRA 50 E 500 PUNTI." :rem 73
410 PRINT"OTTIENI UN % E UNA BOMBA EXTRA
DOPO LE{ 2 SPAZI }ONDATE 5,10,15,20 E
25" :rem 100
420 GOSUB1240:PRINT{CLR}":GOTO480 :rem 138
430 READV:IFV=-1THEN450 :rem 139
440 FORI=VTOV+7:READA:POKEI,A:NEXT:GOTO43
0 :rem 239
450 DIMN(16):FORI=1TO16:READN(I):NEXT
:rem 18
460 FORI=0TO8:READDR(T):NEXT :rem 39
470 RETURN :rem 123
480 PRINT{CLR}":SO=0:TX=0:SC=3072:ZZ=45:
W=1:ME=2:ZA=2 :rem 135
490 DEFFND(Z)=ABS((JV=2)*1+(JV=1)*2+(JV=8
)*3+(JV=4)*4) :rem 205
500 POKETP,1:POKETP-4,1:FORL=1TOZZ:X=INT(
RND(0)*960):POKESC+S+X,5:POKESC+X,27
:rem 115
510 NEXT:ZZ=ZZ+27 :rem 72
520 TR=INT(RND(1)*6)+6:FORP=1TOTR:TA=INT(
RND(1)*960)+SC :rem 61
530 POKETA,38:POKETA+S,2 :rem 205
540 IFINT(RND(1)*100)+1<=3THENPOKETA,35:P
OKETA+S,4 :rem 240
550 NEXT :rem 217
560 SX=INT(RND(1)*960)+SC:POKESX,33:POKES
X+S,6 :rem 73

```



```

570 PO=INT(RND(1)*960)+SC:POKEPO,37:POKEP
O+S,0 :rem 36
580 FORL=1TO3:POKEPO,37:FORP=1TO100:NEXT
:rem 62
590 POKEPO,32:FORP=1TO100:NEXT:NEXT:POKEP
O,37 :rem 160
600 TI$="000000" :rem 248
610 IFTI=>1100THENPOKETP,2:IFTI=>1400THEN
770 :rem 211
620 A=JOY(2):FR=16:IFA>127THENFR=0:A=A-12
8 :rem 216
630 RESTORE 1290:FORI=1TOA:READJV:NEXT:IF
A=0THENJV=0 :rem 103
640 IFJV<>0ANDJV<>128THEN SOUND1,850,1
:rem 10
650 IFFR=0THEN1030 :rem 34
660 XX=FND(Z):IFXX=0THEN710 :rem 185
670 DR=DR(XX):PO=PO+DR:IFPO>4071THENPO=PO
-960:POKEPO-DR+960,32 :rem 139
680 IFPO<3072THENPO=PO+960:POKEPO-DR-960,
32 :rem 112
690 IFPEEK(PO)<>32THEN780 :rem 240
700 POKEPO-DR,32:POKEPO,37:POKEPO+S,0
:rem 57
710 POKESX,32:X=INT(RND(1)*8)+1 :rem 50
720 JJ=DR(X) :rem 169
730 SX=SX+JJ:IFSX<3072THENSX=SX+960:POKES
X-JJ,32 :rem 90
740 IFSX>4071THENSX=SX-960:POKESX-JJ,32
:rem 211
750 IFPEEK(SX)=37THENSX=JJ,33:POKESX+
S-JJ,6:POKEPO,29:POKEPO+S,0:GOTO850
:rem 149
760 POKESX,33:POKESX+S,6:GOTO610:rem 7
770 PRINT"{ 2 GIU' }{BLK}TROPPO LUNGO!":PO
KEPO,29:POKEPO+S,0:GOTO860 :rem 61
780 IFPEEK(PO)=27ORPEEK(PO)=33THENPOKEPO-
DR,29:POKEPO+S-DR,0:GOTO850 :rem 78
790 IFPEEK(PO)<>38THEN820 :rem 242
800 POKEPO-DR,32:POKEPO,37:POKEPO+S,0:SO=
SO+INT(RND(1)*10+1)*50:TX=TX+1
:rem 99
810 IFTX>4THEN950 :rem 19
820 IFPEEK(PO)<>35THENSOUND1,550,10:GOTO7
00 :rem 102
830 POKEPO-DR,32:POKEPO,37:POKEPO+S,0
:rem 61
840 PRINT"{HOME}{ 2 DES }{BLK}3000 PUNTI."
:SO=SO+3000:GOTO950 :rem 133
850 SOUND1,110,30:ME=ME-1:IFME=>1THENZZ=Z
Z-27:SCNCLR:GOTO500 :rem 21
860 PRINT"{HOME}{ 4 GIU' }{BLK}":FORP=1TO1
0 :rem 57
870 PRINT"{BLK}GAME OVER":FORL=1TO100:NEX
T :rem 222
880 PRINT"{SU}{ 13 SPAZI }{SU}":FORL=1TO10
0:NEXT:NEXT:IFSO>HSTHENHS=SO:rem 129
890 POKETP,1:POKETP-4,1:PRINT"{CLR}"TAB(1
2)"{BLK}{ 7 GIU' }PUNTI="SO:PRINTTAB(1
2)"{GIU' }LIV.="W :rem 211
900 PRINTTAB(12)"{GIU' }MAX.PUNTEGGIO="HS
:rem 164
910 PRINTTAB(8)"{ 3 GIU' }{BLK}GIOCHI ANCO
RA (S/N)?" :POKE239,0 :rem 184
920 GETKEY Z$:IFZ$="OR(Z$<>"S"ANDZ$<>"N"
)THEN920 :rem 82
930 IFZ$="S"THENPRINT"{CLR}":GOTO480
:rem 31
940 SCNCLR:END :rem 115
950 BO=INT(1400-TI)*3:IF(PEEK(TP)AND15)=2
THENBO=0 :rem 157

```

```

960 PRINT"{HOME}{ 3 GIU' }{BLK}LIV.":W;"FI
NITO":TX=0:TR=0:IFW=5ORW=10ORW=15ORW=
20ORW=25THEN1000 :rem 191
970 W=W+1:PRINT"{GIU' }BONUS="BO:SO=SO+BO:
PRINT"{GIU' }PUNTI="SO :rem 90
980 PRINT"{GIU' }VITE":ME:PRINT"{GIU' }BOMB
E":ZA :rem 223
990 SOUND1,1000,20:FORI=0TO1000:NEXT:SCNC
LR:GOTO500 :rem 146
1000 FORP=1TO10:PRINT"{HOME}VITA E BOMBA
EXTRA":FORL=1TO100:NEXT :rem 110
1010 PRINT"{SU}{ 17 SPAZI }":FORL=1TO100:N
EXT:NEXT:PRINT"{ 2 GIU' }" :rem 82
1020 ME=ME+1:ZA=ZA+1:GOTO970 :rem 156
1030 IFZA=<0THEN710 :rem 94
1040 ZA=ZA-1:POKETP-4,4:POKETP,3:SOUND3,5
00,50 :rem 24
1050 POKEPO+1,64:POKEPO-1,64:POKEPO-40,93
:POKEPO+40,93 :rem 226
1060 POKEPO+39,78:POKEPO-39,78:POKEPO-41,
77:POKEPO+41,77 :rem 105
1070 POKEPO+1+S,0:POKEPO-1+S,2:POKEPO-40+
S,3:POKEPO+40+S,4 :rem 249
1080 POKEPO+39+S,5:POKEPO-39+S,6:POKEPO-4
1+S,7:POKEPO+41+S,8 :rem 131
1090 FORL=1TO100:NEXT :rem 24
1100 IFPEEK(SX)<>33THENSX=SO+1000:PRINT"
{HOME}{BLK}1000 PUNTI.":GOTO950
:rem 44
1110 POKESX,32:SX=SC+INT(RND(1)*960)+1:PO
KEPO+1,32:POKEPO-1,32 :rem 196
1120 POKEPO-40,32:POKEPO+40,32 :rem 220
1130 POKEPO+39,32:POKEPO-39,32:POKEPO-41,
32:POKEPO+41,32 :rem 65
1140 POKETP,1:POKETP-4,1:TI$="000000":GOT
O710 :rem 104
1150 DATA 14632,60,66,165,129,165,153,66,
60 :rem 82
1160 DATA 14600,195,34,20,156,126,29,8,11
9 :rem 24
1170 DATA 14568,60,66,165,129,153,165,66,
60 :rem 92
1180 DATA 14552,0,28,62,127,127,62,8,8
:rem 83
1190 DATA 14640,6,8,60,126,126,126,60,24
:rem 175
1200 DATA 14616,60,126,255,255,126,60,24,
8,-1 :rem 157
1210 DATA 35,40,45,50,55,60,65,70,75,80,8
5,90,95,100,105,110 :rem 127
1220 DATA 0,40,-40,1,-1,-41,41,-39,39
:rem 239
1230 RETURN :rem 166
1240 PRINTTAB(14)"{ 2 GIU' }{RVS}PREMI UN
TASTO{OFF}": :rem 150
1250 GETKEY Z$: :rem 15
1260 RETURN :rem 169
1270 PRINT"{CLR}MO{ 6 GIU' }{ 2 SIN}T D000
D7FF 3800{ 2 GIU' }{ 12 SIN}X
{ 2 GIU' }{SIN}GOTO160" :rem 73
1280 POKE239,5:POKE1319,19:POKE1320,13:PO
KE1321,13:POKE1322,13:POKE1323,13
:rem 200
1290 DATA1,9,8,10,2,6,4,5 :rem 238

```

Apple Hunt versione per C64

```

100 POKE56,48:CLR:TP=53280 :rem 251
110 FORI=1TO162:READA:X=X+A:NEXT:rem 15
120 IFX<>90689THENPRINT"{CLR}{BLK}ERRORE
NELLE ISTR.DATA11":STOP :rem 178

```



```

130 RESTORE:POKETP,12:POKETP+1,12:PRINT
{CLR}{WHT}{ 9 GIU'}"TAB(10)"BENVENUTI
NELLA" :rem 113
140 PRINTTAB(12)"{ 4 GIU'}{BLK}*{RED}APPL
E{BLU}HUNT{BLK}":GOSUB1300:GOSUB370
:rem 103
150 PRINT{BLK}{ 4 GIU'}"TAB(6)"{RVS}VUOI
LE ISTRUZIONI? (S/N)" :rem 67
160 GETZ$:IFZ$="OR(Z$<"S"ANDZ$<"N")THE
N160 :rem 97
170 IFZ$="N"THEN440 :rem 59
180 POKETP,1:POKETP+1,1 :rem 143
190 A$="{CLR}{BLK}{ 4 GIU'}":PRINTA$TU S
EI %. SCOPO DEL GIOCO E " :rem 165
200 PRINT"MANGIARE { 3 SPAZI}TUTTE LE MELE
({RED}&{BLK})" :rem 79
210 PRINT"NEL BOSCO. DEVI EVITAREGLI ALBE
RI({GRN}){BLK} E " :rem 132
220 PRINT"I MOSTRI({BLU}){BLK}. USA IL J
OY-STICK PER MUOVERE & ATTRAVERSO IL
BOSCO." :rem 105
221 PRINT"PUOI MUOVERTI NELLE QUATTRO DIR
EZIONI." :rem 153
230 PRINT"& HA UN TEMPO LIMITATO PER MANG
IARE{ 5 SPAZI}TUTTE LE {RED}&{BLK}. L
O SCHERMO " :rem 216
240 PRINT"DIVENTA ROSSO SE & STA PER FINI
RE IL SUO TEMPO." :GOSUB1310 :rem 15
250 PRINTA$"{BLU}){BLK} MANGERÀ SIA
{GRN}){BLK} CHE {RED}&{BLK}." :rem 176
260 PRINT"SE & RAGGIUNGE {BLU}){BLK} O
{GRN}){BLK}, HAI PERSO." :rem 156
270 PRINT"PREMI FUOCO PER BUTTARE UNA BOM
BA. UNA{ 2 SPAZI}BOMBA PUO' DISTRUGGE
RE " :rem 180
280 PRINT"{GRN}){BLK} E REINIZIALIZZAI T
EMPO. SE {BLU}){BLK} VIENE COLPITO DA
UNA BOM-BA, " :rem 61
290 PRINT"GUADAGNI 1000 PUNTI." :rem 43
300 PRINT"{BLU}){BLK} VAGHERA' ATTORNO A
{RED}&{BLK} FINO A CHE & LO
{ 2 SPAZI}MANGI O {BLU}){BLK} LO RAGG
IUNGA." :rem 121
305 GOSUB1310 :rem 221
310 PRINTA$"A VOLTE UN {PUR}{BLK} APPARE
SULLO SCHERMO:{ 2 SPAZI}DEVICERCARE
DI " :rem 255
320 PRINT"PRENDERE {PUR}{BLK} PRIMA DI
{BLU}){BLK}. PUOI{ 2 SPAZI}GUADAGNARE
3000 PUNTI SE PRENDI " :rem 2
330 PRINT"{PUR}{BLK}. PER{ 2 SPAZI}OGNI
{RED}&{BLK} PUOI OTTENERE " :rem 168
340 PRINT"TRA 50 E 500 PUNTI." :rem 76
350 PRINT"OTTIENI UN & E UNA BOMBA EXTRA
DOPO LE{ 2 SPAZI}ONDATE 5,10,15,20 E
25" :rem 103
360 GOSUB1310:PRINT{CLR}":GOTO440
:rem 135
370 POKES3272,28:SYS828 :rem 61
380 READV:IFV=-1THEN400 :rem 138
390 FORI=VTOV+7:READA:POKEI,A:NEXT:GOTO38
0 :rem 247
400 S=54272:FORL=STOS+24:POKEL,0:NEXT
:rem 54
410 HF=S+1:DIMN(16):FORI=1TO16:READN(I):N
EXT:GOSUB1340 :rem 68
420 FORT=0TO8:READDR(T):NEXT :rem 35
430 RETURN :rem 119
440 PRINT{CLR}":SO=0:TX=0:SC=1024:ZZ=45:
W=1:ME=2:ZA=2 :rem 126
450 DEFEND(Z)=ABS((JV=2)*1+(JV=1)*2+(JV=8
)*3+(JV=4)*4) :rem 201

```

```

460 POKETP,1:POKETP+1,1:FORL=1TOZZ:X=INT(
RND(0)*960):POKESC+S+X,5:POKES+X,27
:rem 115
470 POKES+24,15:POKES+4,17:POKEHF,50:POKE
S,100:POKES+4,16:NEXT:ZZ=ZZ+27
:rem 138
480 TR=INT(RND(1)*6)+6:FORP=1TOTR:TA=INT(
RND(1)*960)+SC :rem 66
490 POKES+4,65:POKEHF,15:POKES+3,15:POKES
+2,20:POKES+4,64:POKETA,38:POKETA+S,2
:rem 17
500 IFINT(RND(1)*100)+1<=3THENPOKETA,35:P
OKETA+S,4 :rem 236
510 POKES+4,17:POKEHF,50:POKES,200:POKES+
4,16:NEXT :rem 52
520 SX=INT(RND(1)*960)+SC:POKESX,33:POKES
X+S,6 :rem 69
530 PO=INT(RND(1)*960)+SC:POKEPO,37:POKEP
O+S,0 :rem 32
540 FORL=1TO3:POKEPO,37:FORP=1TO100:NEXT
:rem 58
550 POKEPO,32:FORP=1TO100:NEXT:NEXT:POKEP
O,37 :rem 156
560 TI$="000000" :rem 253
570 IFTI=>1100THENPOKETP,2:IFTI=>1400THEN
710 :rem 210
580 JV=PEEK(56320):FR=JVAND16:JV=15-(JVAN
D15):IFFR=0THEN970 :rem 74
590 XX=FND(Z):IFXX=0THEN650 :rem 190
600 DR=DR(XX):PO=PO+DR:IFPO>2023THENPO=PO
-960:POKEPO-DR+960,32 :rem 127
610 IFPO<1024THENPO+960:POKEPO-DR-960,
32 :rem 100
620 IFPEEK(PO)<>32THEN720 :rem 227
630 POKES+4,65:POKEHF,15:POKEPO-DR,32:POK
EPO,37:POKEPO+S,0:POKES+3,15:rem 34
640 POKES+2,20:POKES+4,64 :rem 184
650 POKESX,32:X=INT(RND(1)*8)+1 :rem 53
660 JJ=DR(X) :rem 172
670 SX=SX+JJ:IFSX<1024THENSX=SX+960:POKES
X-JJ,32 :rem 88
680 IFSX>2023THENSX=SX-960:POKESX-JJ,32
:rem 209
690 IFPEEK(SX)=37THENPOKESX-JJ,33:POKESX+
S-JJ,6:POKEPO,29:POKEPO+S,0:GOTO790
:rem 155
700 POKESX,33:POKESX+S,6:GOTO570:rem 6
710 PRINT"{ 2 GIU'}{BLK}TROPO LUNGO!":PO
KEPO,29:POKEPO+S,0:GOTO800 :rem 49
720 IFPEEK(PO)=27ORPEEK(PO)=33THENPOKEPO-
DR,29:POKEPO+S-DR,0:GOTO790 :rem 75
730 IFPEEK(PO)<>38THEN760 :rem 239
740 POKEPO-DR,32:POKEPO,37:POKEPO+S,0:SO=
SO+INT(RND(1)*10+1)*50:TX=TX+1
:rem 102
750 GOSUB1440:IFTX>4THEN890 :rem 156
760 IFPEEK(PO)<>35THEN630 :rem 235
770 POKEPO-DR,32:POKEPO,37:POKEPO+S,0
:rem 64
780 PRINT{HOME}{ 2 DES}{BLK}3000 PUNTI."
:SO=SO+3000:GOTO890 :rem 139
790 GOSUB1390:ME=ME-1:IFME=>1THENZZ=ZZ-27
:PRINT{CLR}":GOTO460 :rem 2
800 PRINT{HOME}{ 4 GIU'}{BLK}":FORP=1TO1
0 :rem 51
810 PRINT{BLK}GAME OVER":FORL=1TO100:NEX
T :rem 216
820 PRINT{SU}{ 13 SPAZI}{SU}":FORL=1TO10
0:NEXT:NEXT:IFSO>HSTHENHS=SO:rem 123
830 POKETP,1:POKETP+1,1:PRINT{CLR}TAB(1
2)"{BLK}{ 7 GIU'}PUNTI="SO:PRINTTAB(1
2)"{GIU'}LIV.="W :rem 200

```



```

840 PRINTTAB(12)"{GIU'}MAX.PUNTEGGIO="HS
      :rem 167
850 PRINTTAB(8)"{ 3 GIU'}{BLK}GIOCHI ANCO
RA (S/N)?":POKE198,0      :rem 191
860 GETZ$:IFZ$="OR(Z$<>"S"ANDZ$<>"N")THE
N860      :rem 111

870 IFZ$="S"THENPRINT"{CLR}":GOTO440
      :rem 30
880 POKE53272,21:END      :rem 114
890 BO=INT(1400-TI)*3:IF(PEEK(TP)AND15)=2
THENBO=0      :rem 160
900 PRINT"{HOME}{ 3 GIU'}{BLK}LIV.":W;"FI
NITO":TX=0:TR=0:IFW=5ORW=10ORW=15ORW=
20ORW=25THEN940      :rem 149
910 W=W+1:PRINT"{GIU'}BONUS="BO:SO=SO+BO:
PRINT"{GIU'}PUNTI="SO      :rem 84
920 PRINT"{GIU'}VITE":ME:PRINT"{GIU'}BOMB
E":ZA      :rem 217
930 GOSUB1340:PRINT"{CLR}":GOTO460
      :rem 143
940 FORP=1TO10:PRINT"{HOME}VITA E BOMBA E
XTRA":FORL=1TO100:NEXT      :rem 74
950 PRINT"{SU}{ 17 SPAZI}":FORL=1TO100:NE
XT:NEXT:PRINT"{ 2 GIU'}"      :rem 46
960 ME=ME+1:ZA=ZA+1:GOTO910      :rem 114
970 IFZA=<0THEN650      :rem 61
980 ZA=ZA-1:POKETP+1,4:POKETP,3      :rem 167
990 POKEPO+1,64:POKEPO-1,64:POKEPO-40,93:
POKEPO+40,93      :rem 190

1000 POKEPO+39,78:POKEPO-39,78:POKEPO-41,
77:POKEPO+41,77      :rem 99
1010 POKEPO+1+S,0:POKEPO-1+S,2:POKEPO-40+
S,3:POKEPO+40+S,4      :rem 243
1020 POKEPO+39+S,5:POKEPO-39+S,6:POKEPO-4
1+S,7:POKEPO+41+S,8      :rem 125
1030 FORL=1TO100:NEXT      :rem 18
1040 IFPEEK(SX)>33THENS0=SO+1000:PRINT"
{HOME}{BLK}1000 PUNTI.":GOTO890
      :rem 50
1050 POKESX,32:SX=SC+INT(RND(1)*960)+1:PO
KEPO+1,32:POKEPO-1,32      :rem 199
1060 POKEPO-40,32:POKEPO+40,32      :rem 223
1070 POKEPO+39,32:POKEPO-39,32:POKEPO-41,
32:POKEPO+41,32      :rem 68

1080 POKETP,1:POKETP+1,1:TI$="000000":GOT
O650      :rem 105
1090 DATA 173,14,220,41,254,141,14
      :rem 123
1100 DATA 220,173,24,208,41,14,10
      :rem 62
1110 DATA 10,133,167,169,208,133,252
      :rem 229
1120 DATA 173,0,221,41,3,73,3      :rem 123
1130 DATA 10,10,10,10,10,10,5      :rem 98

1140 DATA 167,133,254,165,1,41,251
      :rem 130
1150 DATA 133,1,169,0,133,251,133
      :rem 71
1160 DATA 253,168,162,8,177,251,145
      :rem 197
1170 DATA 253,200,208,249,230,252,230
      :rem 19
1180 DATA 254,202,208,242,165,1,9
      :rem 83
1190 DATA 4,133,1,173,14,220,9      :rem 182

```

```

1200 DATA 1,141,14,220,96      :rem 188
1210 DATA 12584,60,66,165,129,165,153,66,
60      :rem 83
1220 DATA12552,195,34,20,156,126,29,8,119
      :rem 25
1230 DATA 12520,60,66,165,129,153,165,66,
60      :rem 75

1240 DATA 12504,0,28,62,127,127,62,8,8
      :rem 75
1250 DATA 12592,6,8,60,126,126,126,60,24
      :rem 176
1260 DATA 12568,60,126,255,255,126,60,24,
8,-1      :rem 167
1270 DATA 35,40,45,50,55,60,65,70,75,80,8
5,90,95,100,105,110      :rem 133

1280 REM DATA 35,63,75,95,107,109,115,125
,131,132,135,140,137,135,137,140
      :rem 38
1290 DATA 0,40,-40,1,-1,-41,41,-39,39
      :rem 246

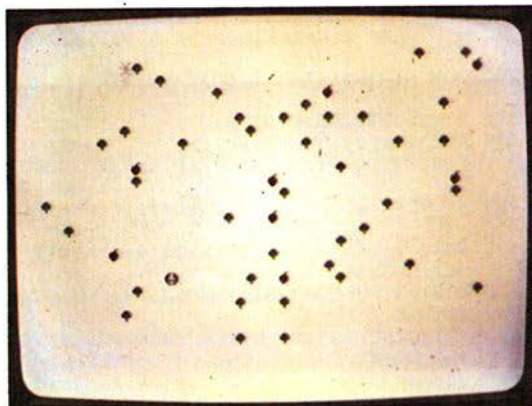
1300 AD=828:FORI=ADTOAD+81:READA:POKEI,A:
NEXTI:RETURN      :rem 162
1310 PRINTTAB(14)"{ 2 GIU'}{RVS}PREMI UN
TASTO{OFF}"      :rem 148
1320 GETZ$:IFZ$=" "THEN1320      :rem 225

1330 RETURN      :rem 167
1340 POKES+24,15:POKES+5,90:POKES+6,240
      :rem 253
1350 POKES+4,17:FORO=0TO3      :rem 241
1360 FORL=1+OTO8+O:POKEHF,N(L)      :rem 72

1370 POKES,110:FORL=1TO50:NEXT:NEXT:NEXT:
FORV=15TO0STEP-1:POKES+24,V:rem 183
1380 FORI=1TO50:NEXT:NEXT:POKES+4,16:RETU
RN      :rem 44
1390 POKES+24,15:POKES+5,90:POKES+6,240
      :rem 2

1400 POKES+4,17      :rem 58
1410 FORL=50TO5STEP-5:POKEHF,L      :rem 128
1420 POKES,5:FORL=1TO50:NEXT:NEXT:FORV=15
TO0STEP-1:POKES+24,V      :rem 221
1430 FORI=1TO50:NEXT:NEXT:POKES+4,16:RETU
RN      :rem 40
1440 POKES+24,15:POKES+4,65:POKES+3,15:PO
KES+2,20:FORI=1TO250STEP5      :rem 246
1450 POKEHF,I:NEXT:POKES+4,64:RETURN
      :rem 64

```





Metabasic per C64

di K. Mykytyn
trad. ed adatt.
di M. Cristulb Grizzi

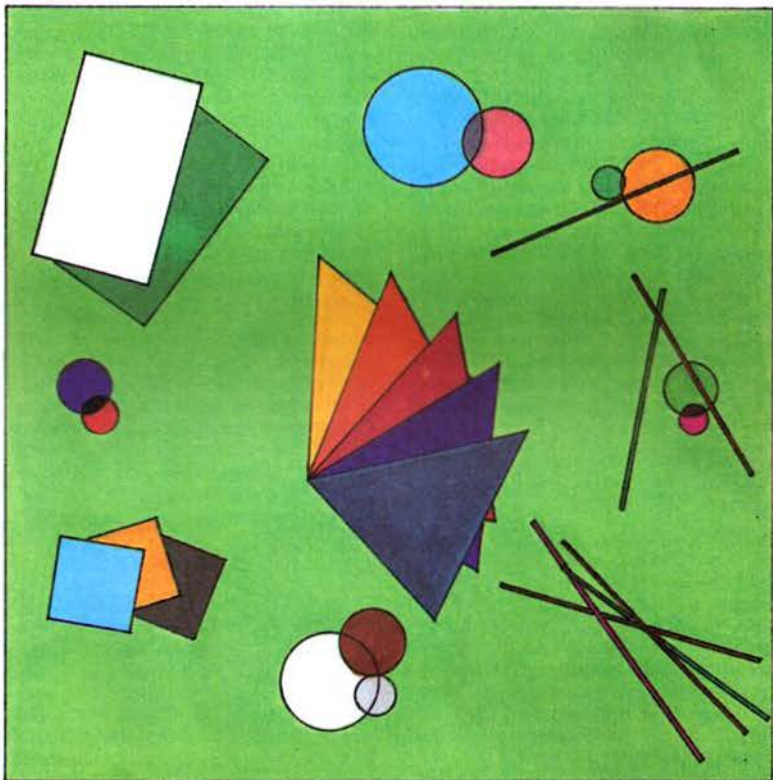
Ecco una poderosa utility che cambierà il vostro modo di programmare. Aggiunge 32 nuovi comandi al BASIC del vostro C64 indirizzati principalmente al "debugging" ed al controllo dei vostri programmi; è utilizzabile sia da sola sia in congiunzione con un monitor/assembler di linguaggio macchina, come il SUPERMONITOR pubblicato sul numero di Gennaio.

Avete acquistato la vostra prima automobile e vi trovate molto bene, ma la prima volta che entrate in autostrada vi accorgete con dispiacere che non supera i 100 chilometri all'ora. Che fare?

Portatela dal vostro meccanico di fiducia, ed egli potrà fornirvi tre alternative: togliere il motore e sostituirlo con un altro; oppure aggiungere al motore che già avete qualche marchingegno turbo-compresso ad iniezione elettronica con waste-gate controllato da microprocessore, oppure ancora, senza aggiungere e togliere nulla, operare una buona messa a punto tramite un'apposita macchina che misura le prestazioni del motore.

Una messa a punto del BASIC

È possibile aggiungere al proprio computer dei nuovi comandi di programmazione in tre modi simili: il primo è disabilitare completamente il BASIC e creare "ex novo" un nuovo linguaggio (un nuovo motore) basato sulle vostre idee circa ciò che dovrebbe permettere un buon linguaggio di programmazione. Il secondo metodo, un'estensione del linguaggio, mantiene il BASIC ed aggiunge alcuni nuovi comandi di programmazione (per suono, grafica in alta risoluzione, o altre funzioni specializzate). Rimane il "motore" BASIC, ma vengono aggiunti alcuni "pezzi", che lo rendono più veloce e più efficiente. Il "Simon's BASIC" e la "Super Expander 64" ne sono degli esempi. Il terzo metodo è come una messa a punto che non modifica il motore. Si



aggiungono comandi da usarsi solo in modo diretto per il "debugging". Non si tratta di un nuovo linguaggio o di un'estensione del BASIC, ma di un cosiddetto "Sistema di sviluppo". I nuovi comandi aggiunti non possono essere usati all'interno di un programma, in quanto funzionano soltanto in modo immediato.

I nuovi linguaggi e le estensioni offrono molteplici vantaggi, ma hanno tutti un grosso lato negativo: si è obbligati a caricare in memoria il linguaggio o l'estensione prima di caricare un programma nel nuovo linguaggio, o che usi i comandi dell'estensione stessa: in caso contrario non funzionerà nulla e si otterranno

messaggi di "?SYNTAX ERROR" a valanga.

Il vantaggio di un sistema di sviluppo come METABASIC risiede nel fatto che l'utility può essere caricata al bisogno, quando si sta scrivendo o verificando un programma. Una volta che questo sarà finito e corretto non si dovrà più caricare METABASIC per mandarlo in esecuzione.

Un'introduzione a METABASIC

METABASIC usa termini mnemonici, così non dovrete ricordare una gran quantità di indirizzi di SYS. Inoltre, se dimenticate la sintassi dei nuovi comandi, potrete o rileggere questo articolo,

o semplicemente digitare **HELP** sulla vostra tastiera.

I programmatori in BASIC disporranno di 12 nuovi comandi. Per la scrittura dei programmi sono disponibili **AUTO**, **KEY** e **UNNEW**. Per esaminare e modificare i programmi ci sono **CHANGE**, **DELETE**, **FIND**, **RENUM** e **VCHANGE**.

Appositamente per il "debugging" sono previsti **DUMP**, **SPEED**, **TRACE** e **TROFF**.

Se siete programmatori in **ASSEMBLY**, potete usare comandi come **MEMORY**, **MONITOR**, **NUMBER** e **@**.

Per controllare **METABASIC** stesso disponete di **DEFAULT**, **HELP**, **INT** e **QUIT**.

I comandi per la gestione del disk drive comprendono **BSAVE**, **CAT**, **DLIST**, **ERR**, **MERGE**, **READ**, **RESAVE**, **SCRATCH**, **SEND** e **START**.

Infine, esiste **LLIST**, se avete una stampante e **TERMINAL**, se avete un modem.

I comandi di METABASIC

Ecco un elenco alfabetico dei nuovi comandi e le istruzioni su come usarli, comprendenti degli esempi. I comandi di **METABASIC** e le stringhe sono stampati in **nero**, mentre i numeri lo sono in **corsivo**. Tutto ciò che è racchiuso tra parentesi è opzionale.

Se qualcosa è descritto come comando per disk drive, non funzionerà se non avete collegato un disk drive. Inoltre, alcuni degli aiuti di programmazione per **L/M** (linguaggio macchina) possono essere anche utili in **BASIC**, e viceversa.

AUTO - Programmazione BASIC

Sintassi: **AUTO** *inizio*, *incremento*

AUTO può eliminare un'operazione molto noiosa quando si scrive un programma.

Genera automaticamente i numeri di linea del programma, partendo dal numero dato come *inizio* ed incrementandolo con il numero dato come *incremento*. Separate i due numeri con una virgola. Dopo aver premuto **RETURN** alla fine di una linea il successivo numero di linea verrà automaticamente generato dal computer. Il numero di linea corrente può essere modificato con l'uso del tasto **INST/DEL** e sostituito da un altro numero. Premere **RUN/STOP** per uscire dal modo **AUTO**.

Esempio:

AUTO 100,10 parte da 100 e continua con 110, 120, 130, ecc.

BSAVE - Comando per disk drive (vedi anche **RESAVE**)

Sintassi: **BSAVE** "*nome file*", *ind. iniziale*, *ind. finale*+1

BSAVE (**Binary SAVE**) salva un'area di memoria su disco, dall'indirizzo iniziale all'indirizzo finale. Inserite il nome del file tra le virgolette, ed usate virgole per

separare nome, indirizzo iniziale ed indirizzo finale. È molto importante aggiungere 1 al valore dell'indirizzo più alto che volete salvare.

Potete usare questo comando per creare copie di programmi in linguaggio macchina, sempre che ne conosciate gli indirizzi iniziale e finale (vedi anche, a questo proposito, la rubrica "Read & Write" di questo numero...). **BSAVE** può essere usato anche per salvare porzioni di memoria di schermo, set di caratteri ridefiniti o schermate in alta risoluzione.

I numeri devono essere in decimale: se desiderate tradurre dei valori da esadecimale in decimale, vedere il comando **NUMBER**.

Dopo aver usato **BSAVE** si può ricaricare il file con **LOAD** "*nome file*", 8,1.

Esempio:

BSAVE "METABASIC", 36864, 40961 per creare su disco una copia di **METABASIC** stesso.

Salvando 100 byte oltre la fine effettiva del programma, salvate anche le definizioni (se precedentemente inserite) dei comandi **DEFAULT** e **KEY** (vedi più avanti). La prossima volta che caricherete il **METABASIC** che avete "Bsalvato" digitate **INT** e **DEFAULT** per riottenerle.

CAT - Comando per disk drive (vedi anche **DLIST**, **READ**)

Sintassi: **CAT**

Ogni volta che desiderate leggere l'intera directory del disco inserito nel drive digitate questo comando. Il programma **BASIC** correntemente in memoria non verrà cancellato. Per leggere solo specifiche parti della directory vedi **DLIST**.

CHANGE - Programmazione BASIC (vedi anche **FIND**, **VCHANGE**)

Sintassi: **CHANGE @** **VECCHIO @** **NUOVO @** (*inizio*, *fine*)

Oppure: **CHANGE @** "**VECCHIO**" **@** "**NUOVO**" **@** (*inizio*, *fine*)

CHANGE cerca nel programma in memoria la stringa definita in "**VECCHIO**" e la sostituisce con quella definita in "**NUOVO**". Le stringhe possono essere lunghe fino a 30 caratteri, e devono essere racchiuse tra due "@". Tutte le linee in cui vengono operate le sostituzioni sono listate sullo schermo.

Il primo tipo di sintassi sostituirà istruzioni **BASIC** e nomi di variabili. Il secondo tipo di sintassi deve essere usato per sostituire le stringhe. I valori "*inizio*" e "*fine*" indicano i numeri di linea da cui deve iniziare e finire la sostituzione; se omessi, questa viene effettuata su tutto il programma. Esempio: **CHANGE @ X @ QQ @ ,1,200** sostituisce al nome di variabile **X** il nome **QQ**, in tutte le linee tra 1 e 200. Le rimanenti linee non vengono modificate.

Per cambiare la stringa "**PIPPA**" in

"**PLUTO**", su tutto il programma, si usa ad esempio:

CHANGE @ "PIPPA" @ "PLUTO" @

DEFAULT - Comando di **METABASIC** (vedi anche **INT**, **QUIT**)

Sintassi: **DEFAULT** *bordo*, *schermo*, *testo*, # *periferica*

Quando si preme **RUN/STOP** + **RESTORE**, lo schermo ritorna ai colori di default, consistenti in caratteri azzurri su schermo blu con bordo azzurro, lo vogliate o no. Inoltre, molti comandi come **LOAD** e **SAVE** assumono il registratore a cassette come numero di periferica di default 1. Il comando **DEFAULT** permette di modificare questi valori in altri che più vi aggradano.

Se usate abitualmente il disk drive, potete cambiare il numero di periferica di default da 1 in 8. Se desiderate usare più spesso il vostro secondo drive, potete usare come numero di periferica di default il 9. Se il vostro C64 è collegato a un televisore in bianco e nero, può risultare utile modificare la combinazione di colori fondo/testo su valori più leggibili.

Nota: non è possibile usare alcuno dei nuovi comandi per il disk drive una volta posto a un valore diverso da 1 il numero di periferica di default. Per disabilitare **DEFAULT**, e ritornare alle condizioni normali, usate il comando **MONITOR** descritto più avanti. Esempio:

DEFAULT 1,1,0,8 regola il colore dello schermo e del bordo sul bianco, il colore del testo sul nero ed il numero di periferica di default su 8. Se premete **RUN/STOP+RESTORE**, vedrete caratteri neri su uno sfondo bianco, e potrete usare, per caricare e salvare su disco, le istruzioni **LOAD** "*nome file*" e **SAVE** "*nome file*" (senza cioè dover aggiungere il ,8).

DELETE - Programmazione BASIC

Sintassi: **DELETE** *inizio*-*fine*

DELETE cancella dalla memoria un certo numero di linee del vostro programma.

Separate il numero della prima linea da cancellare (*inizio*) dall'ultimo (*fine*) con il segno meno (-). Esempio:

DELETE 200-250 cancella tutte le linee tra 200 e 250, estremi compresi.

DLIST - Comando per disk drive (vedi anche **CAT**, **READ**)

Sintassi: **DLIST** "*nome file*"

Questo comando lista sullo schermo un programma **BASIC** contenuto in un disco senza disturbare ciò che è attualmente contenuto in memoria. Il nome del programma deve essere racchiuso tra virgolette. **DLIST** vi permette di controllare il listato di un programma prima di

usare, ad esempio, **MERGE** o **SCRATCH**.

Permette inoltre di leggere porzioni della directory. **DLIST** "\$A" visualizzerà sullo schermo il nome di tutti i file nella directory il cui nome inizia con la lettera A. Esempio:

DLIST "MAGIC HAND" lista sullo schermo il programma "Magic Hand" registrato su disco, senza cancellare il programma BASIC attualmente residente in memoria.

DUMP - Programmazione BASIC

Sintassi: **DUMP**

Potete usare questo comando per esaminare il contenuto di tutte le variabili usate da un programma, esclusi i vettori e le matrici. Date il **RUN** al programma, quindi premete **RUN/STOP** e digitate **DUMP**. Per riprendere l'esecuzione del programma usate **CONT**.

ERR - Comando per disk drive

Sintassi: **ERR**

ERR legge il canale d'errore del drive: usatelo quando la luce rossa del drive sta lampeggiando.

FIND - Programmazione BASIC (vedi anche **CHANGE**, **VCHANGE**)

Sintassi: **FIND @ STRINGA @** (.inizio, fine)

oppure: **FIND @ "STRINGA" @** (.inizio, fine)

Vi permette di trovare una parola, un'istruzione BASIC, una variabile all'interno di un programma. Ogni linea contenente la stringa specificata viene listata sullo schermo. Se desiderate operare la ricerca solo su una parte del programma, specificate in "inizio" e "fine" i numeri di linea da cui iniziare e finire. Se non specificati, la ricerca avviene su tutto il programma. Se volete ricercare istruzioni BASIC, variabili o numeri, usate la prima sintassi. Il secondo tipo di sintassi deve essere usato se si cercano stringhe o caratteri racchiusi tra virgolette. Esempio:

FIND @ A = @ ricerca tutte le linee in cui viene definita la variabile A.

HELP - Comando di METABASIC

Sintassi: **HELP**

Questo comando fornisce una lista completa sullo schermo di tutti i comandi di METABASIC.

INT - Comando di METABASIC (vedi anche **DEFAULT**, **QUIT**)

Sintassi: **INT**

Alcune delle opzioni di METABASIC sono gestite da interrupt. Se si reinizializzano gli interrupt (attraverso il comando

MONITOR), non funzioneranno più i tasti funzione ed il comando **SPEED**. **INT** ripristina la gestione degli interrupt nelle condizioni standard di METABASIC.

KEY - Programmazione BASIC (vedi anche **INT**)

Sintassi: **KEY # funzione, "comando o stringa"**

Questo comando fornisce a METABASIC una grande flessibilità, permettendo di definire ciascuno degli otto tasti funzione con un differente comando o stringa.

Il comando, che può essere lungo fino a dieci caratteri, deve essere racchiuso da virgolette. Esistono inoltre due caratteri speciali: la freccia a sinistra funziona come un **RETURN**, per fare in modo che non dobbiate anche premere **RETURN** dopo la visualizzazione del comando associato al tasto funzione. Inoltre, l'apostrofo (**SHIFT+7**) vale come virgolette. Usando **KEY**, potete disporre di un gran numero di comandi con la semplice pressione di un tasto (come avviene nel BASIC 3.5 del C16-Plus/4). Se desiderate definire in modo permanente i tasti funzione ed i colori dello schermo e del testo, potete usare **KEY** e **DEFAULT** e quindi **BSAVE** "metabasic", usando gli indirizzi iniziale e finale dati a proposito del comando **BSAVE**. Le definizioni verranno salvate insieme al programma. Se gli interrupt vengono accidentalmente modificati, dovete usare il comando **INT** per riabilitare il comando **KEY**. Esempio:

KEY1, "{CLR}LIST100-" pulisce lo schermo e lista il programma a partire dalla linea 100 ogni volta che premete **F1** (la freccia a sinistra indica che il **RETURN** viene dato automaticamente).

KEY7, "DATA" può essere utile con la numerazione automatica (vedi **AUTO**), se si sta scrivendo un programma con molte linee **DATA**. Dopo aver inserito una linea premete **RETURN** e verrà visualizzato il numero di linea successivo. Premete quindi **F7** ed apparirà automaticamente la parola **DATA**.

KEY2, "VERIFY" definisce **F2** in modo che stampi **VERIFY** più un **RETURN** (si noti che gli apostrofi sono stati cambiati in virgolette). Se avete usato **DEFAULT** per cambiare il numero di periferica di default in 8, la pressione di **F2** provocherà la verifica automatica del programma salvato più recentemente sul disco.

LLIST - Comando per la stampante

Sintassi: **LLIST** inizio-fine

Questo comando lista un programma, ma il listato viene fatto sulla stampante invece che sullo schermo. I numeri di linea "inizio" e "fine" sono opzionali. La sintassi di **LLIST** è identica a quella di **LIST**. Esempio:

LLIST10-20 lista su stampante le li-

nee di programma tra 10 e 20, estremi compresi.

MEMORY - Programmazione L/M (vedi anche **@**)

Sintassi: **MEMORY** ind.iniziale-(ind. finale)

È possibile esaminare, con questo comando, qualsiasi parte della memoria. Usate numeri decimali (non esadecimali!) per gli indirizzi iniziale e finale. I valori contenuti in memoria vengono visualizzati, sei byte per linea, in decimale. Inoltre, l'equivalente in caratteri ASCII viene visualizzato in campo inverso nella parte destra dello schermo. Questo comando ha un effetto molto simile al comando **M** di **SUPERMONITOR**, con la differenza che **MEMORY** lavora in decimale. Se non esiste un corrispondente carattere ASCII, viene visualizzato un punto.

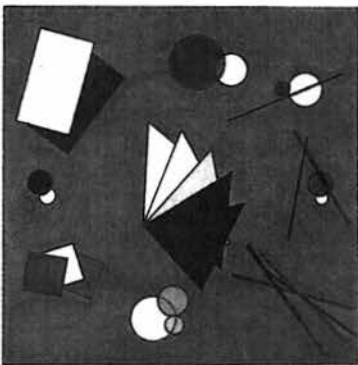
Se si omette l'indirizzo finale, ad esempio con **MEMORY43**, verrà visualizzato il contenuto di due byte (43 e 44). Ciò rende facile leggere il contenuto dei puntatori a due byte. Per modificare il contenuto della memoria si può usare il comando **@**, come descritto più avanti. Esempio:

MEMORY41374-41474 visualizza alcuni dei primi messaggi d'errore (si noti che il valore ASCII dell'ultimo carattere è sempre sommato a 128). Oppure, caricate un programma BASIC ed usate: **MEMORY2048-2148** per vedere come quest'ultimo è immagazzinato in memoria.

MERGE - Comando per disk drive

Sintassi: **MERGE** "nome file"

MERGE legge un programma da disco, lista ogni sua linea sullo schermo e la aggiunge al programma attualmente in memoria. Se i due programmi possiedono numeri di linea comuni, il programma su disco ha la precedenza. Poniamo che entrambi i programmi posseggano



la linea 250: la linea 250 del programma su disco rimpiazzerà la linea 250 del programma in memoria.

Prima di usare questo comando può essere utile usare **DLIST**, per assicurarsi di operare il **MERGE** del programma corretto. Se ci sono conflitti nei numeri di linea, si può usare **RENUM** (vedi più avanti) per rinumerare uno dei due programmi. Se desiderate fare il **MERGE** di una sola parte di programma, usate **DELETE** per cancellare la parte che non interessa.

MONITOR - Programmazione L/M

Sintassi: **MONITOR**

Se avete in memoria un monitor in linguaggio macchina (ad esempio il **SUPERMONITOR**), potete entrarvi con questo comando (ammesso che esso venga abilitato tramite un **BRK**). Per usare **METABASIC** con un monitor dovete caricare **METABASIC**, digitare **NEW** e dare la **SYS36864**. Quindi caricare il monitor, digitare **NEW** e dare la **SYS** di attivazione del monitor (49152 per il **SUPERMONITOR**), che inizierà il vettore **BKR**, facendo in modo che punti al monitor.

MONITOR serve anche ad altre operazioni: riporta i colori di testo, bordo e schermo ai valori di default (blu ed azzurro) e riporta allo standard l'interrupt, disabilitando quindi le definizioni dei tasti funzione ed il comando **SPEED**. Essi si possono comunque ripristinare con il comando **INT**.

NUMBER - Programmazione L/M

Sintassi: **NUMBER\$esadecimale**

oppure: **NUMBERdecimale**

NUMBER permette di convertire numeri da base decimale in esadecimale, e viceversa. I numeri esadecimali devono essere preceduti dal segno \$. Inoltre, il

numero viene convertito nel formato byte basso/byte alto (in decimale) e nel numero binario equivalente (contrassegnato dal segno %). Esempio:

NUMBER\$100

256

0 1

%100000000

NUMBER34

\$22

34 0

%100010

QUIT - Comando di METABASIC

Sintassi: **QUIT**

Reinizializza tutti i vettori e disabilita tutti i comandi di **METABASIC**. Non ripristina tuttavia il puntatore di fine memoria, lasciando **METABASIC** ancora protetto dal **BASIC**. Si può riattivare **METABASIC** con **SYS36864**, oppure con **SYS94096**.

READ - Comando per disk drive (vedi anche CAT, DLIST)

Sintassi: **READ"nome file seq"**

READ permette di esaminare il contenuto di file sequenziali su disco. Il contenuto del file viene visualizzato sullo schermo senza alterare il contenuto della memoria. Nel raro caso in cui vorreste usare il comando **BASIC READ** (per leggere istruzioni **DATA**) in modo diretto dovreste farlo precedere dai due punti.

RENUM - Programmazione BASIC

Sintassi: **RENUM(inizio) (incremento)**

L'intero programma **BASIC** in memoria viene rinumerato, partendo dal numero di linea eventualmente specificato in "inizio", con incremento eventualmente specificato in "incremento". Se si omettono questi valori, la rinumerazione parte dalla linea 10 con incrementi di 10. Questo comando rinumererà anche i **GO TO**, **GOSUB**, **ON-GOTO**, **ON-GOSUB**, **IF-THEN**, ecc.

Un cenno va fatto al **GO TO** (con uno spazio tra **GO TO**), che non viene rinumerato. Usate **FIND** per vedere se esiste qualche **GO TO** nel vostro programma, e trasformatelo in **GOTO** prima di usare **RENUM**. Esempio:

RENUM100,20 rinumererà il programma a partire da 100 con incrementi di 20.

RESAVE - Comando per disk drive

Sintassi: **RESAVE"nome file"**

È un "Save con replace" il quale risolve tutti i problemi che a volte si incontrano con il solito comando **SAVE @:nome file**. Quest'ultimo, infatti, prima salva

il nuovo file e poi cancella il precedente, in modo che — se non vi è spazio sufficiente sul disco — si possono avere dei problemi. Il classico "Save con replace" è a volte inaffidabile e può provocare la perdita di entrambi i file e, anche se alcuni esperti non sono d'accordo su questa affermazione, è quindi da evitare il più possibile.

RESAVE, al contrario, cancella il vecchio file prima di salvare il nuovo, ed è quindi scevro da tutti i rischi insiti nel **SAVE @:nome file**: può essere usato con la massima tranquillità.

SCRATCH - Comando per disk drive

Sintassi: **SCRATCH"nome file"**

SCRATCH esegue la stessa funzione della solita **OPEN15,8,15:PRINT # 15, "S:nome file":CLOSE15**, con un singolo comando. Cancellando quindi un file dal disco. Se avete appena inserito un nuovo disco non drive, non è una cattiva idea iniziarlo (vedi **SEND**) prima di cancellare un file. Potete usare l'asterisco per cancellare più di un programma: **SCRATCH"A"** cancellerà tutti i file il cui nome inizia per A. Esempio: **SCRATCH"MAGIC HAND"** cancella dal disco il file chiamato "Magic Hand".

SEND - Comando per disk drive

Sintassi: **SEND"comando"**

È un modo molto conveniente per inviare comandi al drive attraverso il canale 15. **SEND"I"** inizializza il drive, **SEND"V"** esegue una validate, **SEND"R:nuovo=vecchio"** cambia il nome ad un file e così via. Per maggiori dettagli fare riferimento al manuale del drive od alla serie di articoli "Iniziare con un sistema a dischi" da noi precedentemente pubblicata.

SPEED - Programmazione BASIC

Sintassi: **SPEEDnumero**

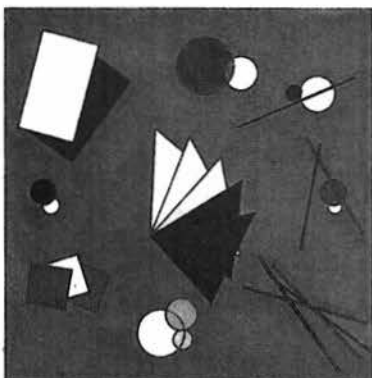
SPEED seguito da un numero tra 0 e 255 modifica la velocità di stampa. Più alto è il numero, più bassa è la velocità. Provate a digitare **SPEED255** e quindi listate un programma. Potete ritornare in condizioni normali con **SPEED0**.

Se questo comando non funziona, usate prima **INT** per ripristinare correttamente gli interrupt. **SPEED** è molto utile anche nell'uso del comando **TRACE** (vedi più avanti).

START - Comando per disk drive

Sintassi: **START"nome file"**

Se vi siete dimenticati l'indirizzo iniziale





Jackson, naturalmente.

**ORDINE
MINIMO
L. 50.000**

Pagina 144

di un programma in linguaggio macchina che avete su disco (cioè il numero da dare nella SYS di attivazione), questo comando lo visualizza. Esempio: **START"METABASIC"** provoca la visualizzazione sullo schermo del numero 36864.

TERMINAL - Comando per modem

Sintassi: **TERMINAL**

Se disponete di un modem Commodore e lo avete collegato al C64, il comando **TERMINAL** trasforma il vostro computer in un "terminale stupido" a 300 baud che potete usare per comunicare con servizi di informazione che usano l'ASCII standard. Non potete, con questo comando, modificare alcuno dei parametri di default (come il full duplex), né potete caricare in memoria o salvare via modem testi o programmi: potete semplicemente visualizzarli sul video. Per ritornare al BASIC premete il tasto **£** (simbolo della Lira); non premete **RUN/STOP+RESTORE**. Una cautela: le locazioni di memoria da 52736 a 53247 vengono usate come buffer, quindi qualunque programma che risiede in quest'area verrà sovrascritto.

TRACE - Programmazione BASIC (vedi anche TROFF)

Sintassi: **TRACE** seguito da **RUN**

Se state cercando qualche errore in un programma BASIC, **TRACE** è molto utile. Man mano che una linea viene eseguita il suo numero viene visualizzato sullo schermo. Usate **SHIFT** o **CTRL** per fermare temporaneamente l'esecuzione del programma. **SPEED** controlla la velocità di esecuzione e **TROFF** disabilita il **TRACE**.

TROFF - Programmazione BASIC

Sintassi: **TROFF**

Disabilita la funzione **TRACE**.

UNNEW - Programmazione BASIC

Sintassi: **UNNEW**

Se accidentalmente digitate **NEW** e perdete un programma, potrete recuperarlo digitando **UNNEW**.

VCHANGE - Programmazione BASIC (vedi anche **CHANGE**, **FIND**)

Sintassi: **VCHANGE @ VECCHIO @ NUOVO @** (inizio, fine)

oppure: **VCHANGE @ "VECCHIO" @ "NUOVO" @** (inizio, fine)

VCHANGE (Verify **CHANGE**) funziona

come **CHANGE**, con l'eccezione che vi viene data possibilità di decidere, caso per caso, se effettuare o meno il cambiamento.

Ogni linea contenente la vecchia stringa viene visualizzata, con ogni occorrenza della stringa evidenziata da un cerchietto pieno. Se premete **Y**, il cambiamento viene effettuato. Se premete **N**, si passa alla linea successiva.

@ - Programmazione L/M

Sintassi: **@ ind.iniziale, numero, numero,...**

Funziona come una **POKE**, con la differenza che permette di inserire una serie di numeri in locazioni di memoria consecutive. Ad esempio, se volete portare i colori di bordo e sfondo in bianco, potrete usare **@53280,1,1**. Il primo 1 viene inserito in 53280, mentre il secondo in 53281. Se aggiungete altri numeri, separati da virgole, verranno immessi nelle locazioni 53282, 53283 e così via...

Questo comando può inoltre essere utilizzato insieme a **MEMORY**. Per prima cosa fate l'equivalente di una **PEEK** in una serie di locazioni di memoria con **MEMORY**.

Quindi modificate il contenuto di queste locazioni, ponendo una **@** all'inizio di ogni linea di byte che volete modificare. Andate con il cursore sui numeri che volete modificare, modificateli e premete **RETURN**.

Note per il caricamento o la digitazione di METABASIC

Se si dispone della cassetta allegata alla rivista, è sufficiente posizionarsi con il nastro in corrispondenza dell'inizio del programma e digitare:

LOAD"" ,1,1

(attenzione: non inserire spazi tra gli apici!).

Il programma andrà quindi mandato in esecuzione, digitando:

**NEW
SYS36864**

Ricordiamo che per conoscere il punto esatto di inizio dei vari programmi su cassetta è sufficiente inserire quest'ultima completamente riavvolta nel registratore, azzerare il contanastro e digitare:

LOAD""

Dal momento che sicuramente non esiste sulla cassetta alcun programma di nome "", il computer passerà in rassegna tutti i programmi incontrati, visualizzandone il nome con il messaggio:

FOUND XXXX

Sarà quindi utile annotarsi su un foglio i nomi dei programmi e la relativa posizione del contanastro (sottrarre tre o quattro unità dal valore corrispondente all'apparizione del messaggio **FOUND**...).

Gli abbonati alla sola rivista dovranno invece usare il programma **MLX** (vedere nelle pagine verdi) per digitare il listato in linguaggio macchina. La procedura da seguire, in questo caso, è:

1. Accendere il computer e digitare:

POKE644,144:SYS58260 RETURN

Caricare in memoria **MLX** (che sarà stato precedentemente digitato e salvato), e rispondere alle sue richieste circa gli indirizzi iniziale e finale con:

indirizzo iniziale: 36864

indirizzo finale: 40805

Dopo avere digitato e salvato il programma tramite **MLX** si potranno seguire per il caricamento le istruzioni date più sopra per i possessori di cassetta.

Note tecniche

Il programma usa gli ultimi 4 Kbyte della memoria **BASIC** e vi lascia quindi liberi per i normali programmi circa 35 Kbyte di memoria. La prima cosa eseguita da **METABASIC** è lo spostamento in basso del puntatore di fine memoria, per proteggere se stesso dalle variabili. Dopo la **SYS** di attivazione può sembrare che non accada nulla di particolare: in realtà, **METABASIC** sta "lavorando" in sottofondo e vi trovate ad avere i nuovi 32 comandi prima descritti.

Digitate sempre **NEW** dopo aver caricato **METABASIC**. Una caratteristica che funziona in modo automatico è la pausa nel **LIST**. Quando state listando un programma tenete premuti **CTRL**, **SHIFT** o il tasto Commodore per bloccare temporaneamente il listing.

Se desiderate interrompere l'esecuzione dei vari comandi come ad esempio **RENUM**, usate solo **RUN/STOP**, non **RUN/STOP+RESTORE**!

I vari comandi funzionano solo in modo diretto: non è possibile inserirli nei vostri programmi. Al contrario dei comandi **BASIC** ordinari, non sono ammesse abbreviazioni.

I programmatori in **L/M** devono tener presente che **METABASIC** occupa le locazioni da \$9000 a \$9FFF. Il segmento da 4 Kbyte a partire da \$C000 è quindi interamente disponibile per altri programmi in **L/M**.

Infine, assicuratevi di caricare **METABASIC** prima di qualunque altro programma, sia esso in **BASIC** od in **L/M**.

Metabasic per C64

36864 :076,194,145,000,000,000,159
 36870 :000,000,000,000,000,000,006
 36876 :000,000,063,000,000,000,075
 36882 :199,008,000,000,000,066,035
 36888 :006,000,000,004,000,008,042
 36894 :000,000,000,000,000,000,030
 36900 :000,169,001,044,169,000,163
 36906 :141,036,144,032,146,145,174
 36912 :169,069,141,040,003,169,127
 36918 :145,141,041,003,173,029,074
 36924 :144,133,183,134,187,132,205
 36930 :188,160,002,169,000,153,226
 36936 :086,002,136,016,250,169,219
 36942 :008,133,186,169,096,133,035
 36948 :185,032,108,144,144,021,206
 36954 :169,002,032,195,255,169,144
 36960 :015,032,195,255,032,231,088
 36966 :255,162,005,108,000,003,123
 36972 :032,213,243,165,186,032,211
 36978 :180,255,165,185,032,150,057
 36984 :255,169,152,141,000,003,072
 36990 :169,144,141,001,003,169,241
 36996 :044,141,002,003,169,145,124
 37002 :141,003,003,032,165,255,225
 37008 :032,080,145,176,072,032,169
 37014 :165,255,032,165,255,141,139
 37020 :023,144,032,165,255,013,020
 37026 :023,144,240,055,032,165,053
 37032 :255,133,020,141,023,144,116
 37038 :032,165,255,133,021,141,153
 37044 :024,144,173,036,144,208,141
 37050 :004,160,001,208,002,160,209
 37056 :255,200,032,165,255,153,228
 37062 :000,002,208,247,032,060,235
 37068 :145,032,137,145,152,024,071
 37074 :105,005,168,173,036,144,073
 37080 :240,045,076,162,164,032,167
 37086 :171,255,032,066,246,032,000
 37092 :231,255,169,131,141,002,133
 37098 :003,169,164,141,003,003,205
 37104 :169,139,141,000,003,169,093
 37110 :227,141,001,003,169,237,000
 37116 :141,040,003,169,246,141,224
 37122 :041,003,076,055,158,032,111
 37128 :033,159,174,023,144,173,202
 37134 :024,144,032,205,189,169,009
 37140 :000,133,095,169,002,133,040
 37146 :096,169,000,133,073,169,154
 37152 :086,141,000,002,169,002,176
 37158 :141,001,002,076,237,166,149
 37164 :173,023,144,133,020,173,198
 37170 :024,144,133,021,032,019,167
 37176 :166,076,201,166,072,165,134
 37182 :203,201,063,240,154,104,003
 37188 :096,072,165,203,201,063,100
 37194 :240,145,104,076,237,246,098
 37200 :141,018,144,024,165,144,204

37206 :240,045,032,033,159,169,252
 37212 :015,162,008,168,032,186,151
 37218 :255,169,000,032,189,255,230
 37224 :032,192,255,162,015,032,024
 37230 :198,255,032,207,255,072,105
 37236 :032,210,255,104,201,013,163
 37242 :208,244,169,015,032,195,217
 37248 :255,032,204,255,056,173,079
 37254 :018,144,096,160,255,200,239
 37260 :185,000,002,208,250,096,113
 37266 :032,121,000,201,000,240,228
 37272 :039,032,115,000,144,008,234
 37278 :201,034,240,004,056,076,001
 37284 :181,145,032,158,173,165,250
 37290 :013,208,009,032,155,188,007
 37296 :165,100,164,101,024,096,058
 37302 :032,130,183,141,029,144,073
 37308 :166,034,164,035,056,096,227
 37314 :120,169,235,141,004,003,098
 37320 :169,145,141,005,003,169,064
 37326 :065,141,020,003,169,147,239
 37332 :141,021,003,032,062,148,107
 37338 :088,165,056,201,144,144,248
 37344 :007,169,144,133,056,032,253
 37350 :092,166,076,055,158,162,171
 37356 :255,232,189,000,002,201,091
 37362 :032,240,248,160,255,141,038
 37368 :023,144,200,192,224,176,183
 37374 :079,217,081,146,208,024,241
 37380 :140,024,144,142,025,144,111
 37386 :232,200,185,081,146,048,134
 37392 :029,221,000,002,240,244,240
 37398 :172,024,144,174,025,144,193
 37404 :200,192,224,176,045,185,026
 37410 :081,146,016,246,173,023,207
 37416 :144,200,200,076,250,145,031
 37422 :142,027,144,134,122,232,079
 37428 :189,000,002,240,004,201,176
 37434 :065,176,017,200,185,081,014
 37440 :146,141,076,146,200,185,190
 37446 :081,146,141,077,146,076,225
 37452 :039,159,076,124,165,065,192
 37458 :085,084,207,021,152,066,185
 37464 :083,065,086,197,039,159,205
 37470 :067,065,212,047,147,067,187
 37476 :072,065,078,071,197,212,027
 37482 :155,068,069,070,065,085,106
 37488 :076,212,242,147,068,069,158
 37494 :076,069,084,197,182,148,106
 37500 :068,076,073,083,212,040,164
 37506 :144,068,085,077,208,087,031
 37512 :151,069,082,210,056,148,084
 37518 :070,073,078,196,200,155,146
 37524 :072,069,076,208,238,157,200
 37530 :073,078,212,194,145,075,163
 37536 :069,217,149,147,076,076,126
 37542 :073,083,212,144,157,077,144
 37548 :069,077,079,082,217,057,241
 37554 :149,077,069,082,071,197,055

37560 :037,144,077,079,078,073,160
 37566 :084,079,210,004,150,078,027
 37572 :085,077,066,069,210,166,101
 37578 :150,081,085,073,212,001,036
 37584 :150,082,069,065,196,103,105
 37590 :148,082,069,078,085,205,113
 37596 :000,153,082,069,083,065,160
 37602 :086,197,014,150,083,067,055
 37608 :082,065,084,067,200,005,223
 37614 :150,083,069,078,196,025,071
 37620 :150,083,080,069,069,196,123
 37626 :229,157,083,084,065,082,182
 37632 :212,106,148,084,069,082,189
 37638 :077,073,078,065,204,064,055
 37644 :158,084,082,065,067,197,153
 37650 :174,152,084,082,079,070,147
 37656 :198,243,152,085,078,078,090
 37662 :069,215,155,150,086,067,004
 37668 :072,065,078,071,197,188,195
 37674 :155,064,160,208,147,160,168
 37680 :001,185,061,147,145,122,197
 37686 :200,192,004,208,246,076,212
 37692 :040,144,034,036,000,165,223
 37698 :203,205,014,144,240,074,178
 37704 :141,014,144,201,003,208,015
 37710 :004,169,007,208,008,201,163
 37716 :004,144,059,201,007,176,163
 37722 :055,056,233,004,010,010,202
 37728 :010,010,010,174,141,002,187
 37734 :240,003,024,105,016,168,146
 37740 :162,000,185,097,159,201,144
 37746 :095,208,002,169,013,201,034
 37752 :039,208,008,165,212,073,057
 37758 :001,133,212,169,034,201,108
 37764 :000,240,009,157,119,002,147
 37770 :232,200,224,010,208,222,210
 37776 :134,198,076,049,234,032,099
 37782 :146,145,136,152,048,047,056
 37788 :201,008,176,043,010,010,092
 37794 :010,010,024,105,097,133,029
 37800 :251,169,159,105,000,133,217
 37806 :252,032,146,145,144,023,148
 37812 :173,029,144,201,011,176,146
 37818 :016,168,169,000,145,251,167
 37824 :136,177,034,145,251,136,047
 37830 :016,249,076,055,158,162,146
 37836 :011,108,000,003,169,255,238
 37842 :133,251,032,166,145,176,089
 37848 :242,140,234,147,141,235,075
 37854 :147,032,146,145,176,011,111
 37860 :152,230,251,164,251,153,149
 37866 :000,000,076,223,147,108,020
 37872 :002,003,120,169,095,141,002
 37878 :024,003,169,157,141,025,253
 37884 :003,088,169,003,133,251,131
 37890 :032,146,145,176,010,152,151
 37896 :166,251,157,019,144,198,175
 37902 :251,016,241,169,040,141,104
 37908 :050,003,169,148,141,051,070

LOAD

Metabasic

37914 :003,169,048,141,048,003,182
 37920 :169,148,141,049,003,076,106
 37926 :055,158,174,019,144,134,210
 37932 :186,076,237,245,174,019,213
 37938 :144,134,186,076,165,244,231
 37944 :032,088,145,076,055,158,098
 37950 :169,073,141,038,003,169,143
 37956 :148,141,039,003,096,072,055
 37962 :138,072,152,072,173,141,054
 37968 :002,208,251,174,012,144,103
 37974 :232,160,001,136,208,253,052
 37980 :202,208,250,104,168,104,104
 37986 :170,104,076,202,241,169,036
 37992 :001,044,169,000,141,011,214
 37998 :144,032,146,145,032,189,030
 38004 :255,169,002,162,008,160,104
 38010 :002,032,186,255,032,192,053
 38016 :255,144,003,076,090,144,072
 38022 :162,002,032,198,255,032,047
 38028 :165,255,032,080,145,176,225
 38034 :024,174,011,144,208,010,205
 38040 :170,032,165,255,032,205,243
 38046 :189,076,171,148,032,210,216
 38052 :255,032,060,145,076,139,103
 38058 :148,169,002,032,195,255,203
 38064 :032,204,255,076,055,158,188
 38070 :032,146,145,176,115,132,160
 38076 :253,133,254,032,049,149,034
 38082 :160,002,177,095,141,023,024
 38088 :144,200,177,095,141,024,213
 38094 :144,165,095,133,251,165,135
 38100 :096,133,252,032,146,145,248
 38106 :176,082,200,208,003,024,143
 38112 :105,001,032,049,149,056,104
 38118 :165,101,229,253,165,100,219
 38124 :229,254,144,062,160,000,061
 38130 :140,000,002,165,095,145,021
 38136 :251,200,165,096,145,251,076
 38142 :173,023,144,133,020,173,152
 38148 :024,144,133,021,160,005,235
 38154 :173,002,003,133,251,173,233
 38160 :003,003,133,252,169,033,097
 38166 :141,002,003,169,149,141,115
 38172 :003,003,076,162,164,165,089
 38178 :251,141,002,003,165,252,080
 38184 :141,003,003,076,049,158,214
 38190 :076,203,147,132,020,133,245
 38196 :021,032,019,166,096,032,162
 38202 :146,145,132,251,133,252,093

38208	:032,146,145,132,253,133,137	38562	:165,076,049,158,032,146,020
38214	:254,032,184,149,076,116,113	38568	:145,176,050,132,251,133,031
38220	:149,032,060,145,056,165,171	38574	:252,072,169,036,032,210,177
38226	:253,229,251,165,254,229,183	38580	:255,104,032,192,150,152,041
38232	:252,176,010,173,028,144,103	38586	:032,192,150,076,240,150,002
38238	:201,001,208,003,076,055,126	38592	:072,074,074,074,074,032,080
38244	:158,238,028,144,173,028,101	38598	:207,150,104,041,015,032,235
38250	:144,201,006,208,005,169,071	38604	:207,150,096,201,010,144,244
38256	:001,032,147,149,173,028,130	38610	:003,024,105,007,024,105,222
38262	:144,010,010,024,105,008,163	38616	:048,032,210,255,096,169,002
38268	:133,211,160,000,177,251,032	38622	:000,133,251,133,252,168,135
38274	:032,210,149,169,058,032,012	38628	:200,177,122,208,071,165,147
38280	:210,255,230,251,208,002,012	38634	:252,166,251,032,205,189,049
38286	:230,252,076,077,149,169,071	38640	:032,033,159,169,000,166,031
38292	:058,032,210,255,169,018,122	38646	:251,032,205,189,169,032,100
38298	:032,210,255,198,252,160,237	38652	:032,210,255,169,000,166,060
38304	:250,177,251,201,032,144,191	38658	:252,032,205,189,032,033,233
38310	:007,201,127,176,003,076,244	38664	:159,169,037,032,210,255,102
38316	:176,149,169,046,032,210,186	38670	:169,000,141,028,144,162,146
38322	:255,200,208,235,230,252,022	38676	:016,169,048,006,251,038,036
38328	:032,033,159,169,032,032,129	38682	:252,144,005,169,049,238,115
38334	:210,255,165,252,166,251,209	38688	:028,144,172,028,144,240,020
38340	:032,205,189,169,000,141,164	38694	:003,032,210,255,202,208,180
38346	:028,144,169,062,032,210,079	38700	:232,076,055,158,162,004,219
38352	:255,096,168,032,162,179,076	38706	:006,251,038,252,202,208,239
38358	:032,221,189,160,255,200,247	38712	:249,032,074,151,024,101,175
38364	:185,000,001,208,250,140,236	38718	:251,133,251,165,252,105,195
38370	:023,144,136,136,136,185,218	38724	:000,133,252,076,228,150,139
38376	:000,001,201,049,144,007,122	38730	:056,233,048,201,010,144,254
38382	:201,058,176,003,076,247,231	38736	:002,233,007,096,083,048,037
38388	:149,169,048,032,210,255,083	38742	:058,032,088,159,234,165,054
38394	:200,204,023,144,208,231,236	38748	:046,133,252,076,116,151,098
38400	:096,032,083,228,000,169,096	38754	:169,058,032,210,255,165,219
38406	:000,141,017,144,133,180,109	38760	:251,024,105,007,133,251,107
38412	:240,020,169,000,141,017,087	38766	:165,252,105,000,133,252,249
38418	:144,169,001,133,180,208,085	38772	:160,000,056,165,251,229,209
38424	:009,169,001,141,017,144,249	38778	:047,165,252,229,048,144,239
38430	:169,000,133,180,032,146,178	38784	:005,169,000,076,049,168,083
38436	:145,172,029,144,169,000,183	38790	:169,000,141,028,144,177,025
38442	:153,003,002,136,177,034,035	38796	:251,016,004,056,046,028,029
38448	:153,003,002,136,016,248,094	38802	:144,041,127,032,210,255,187
38454	:173,017,144,208,011,160,255	38808	:200,177,251,016,004,056,088
38460	:002,185,084,151,153,000,123	38814	:046,028,144,041,127,240,016
38466	:002,136,016,247,169,015,139	38820	:003,032,210,255,173,028,097
38472	:162,008,160,015,032,186,123	38826	:144,208,024,032,004,152,222
38478	:255,032,137,145,152,174,205	38832	:200,152,024,101,251,072,208
38484	:017,144,240,009,056,233,015	38838	:165,252,105,000,168,104,208
38490	:003,162,003,160,002,208,116	38844	:032,162,187,032,010,152,251
38496	:004,162,000,160,002,032,200	38850	:076,098,151,201,003,208,163
38502	:189,255,032,192,255,144,145	38856	:026,169,037,032,210,255,161
38508	:003,076,090,144,169,015,093	38862	:032,004,152,200,177,251,254
38514	:032,195,255,165,180,240,157	38868	:072,200,177,251,168,104,160
38520	:028,169,003,162,008,160,138	38874	:032,145,179,032,010,152,000
38526	:003,032,186,255,173,029,036	38880	:076,098,151,201,001,208,191
38532	:144,162,003,160,002,032,123	38886	:026,169,036,032,210,255,190
38538	:189,255,169,043,166,045,237	38892	:032,004,152,200,177,251,028
38544	:164,046,032,216,255,032,121	38898	:072,200,177,251,133,034,085
38550	:080,145,076,055,158,160,056	38904	:200,177,251,133,035,104,124
38556	:001,152,145,043,032,051,068	38910	:032,036,171,076,098,151,050



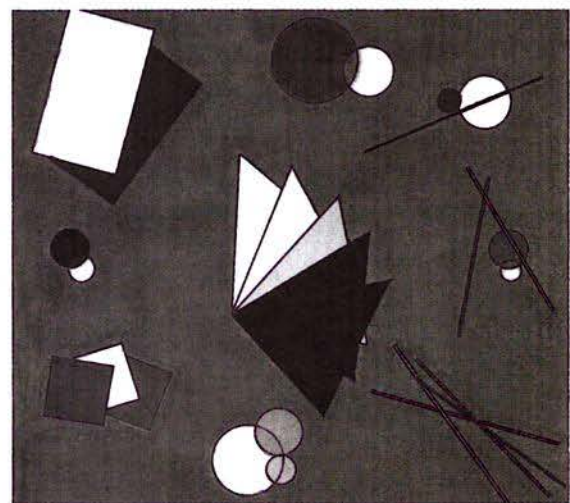
38916 :169,061,032,210,255,096,059
 38922 :032,221,189,072,152,170,078
 38928 :104,032,030,171,096,032,225
 38934 :146,145,140,034,144,141,004
 38940 :035,144,032,146,145,140,158
 38946 :032,144,141,033,144,173,189
 38952 :034,144,056,237,032,144,175
 38958 :141,034,144,173,035,144,205
 38964 :237,033,144,141,035,144,018
 38970 :173,034,144,133,020,173,223
 38976 :035,144,133,021,120,169,174
 38982 :093,141,002,003,169,152,118
 38988 :141,003,003,169,141,141,162
 38994 :020,003,169,152,141,021,076
 39000 :003,088,076,055,158,165,121
 39006 :020,024,109,032,144,133,044
 39012 :099,165,021,109,033,144,159
 39018 :133,098,162,144,056,032,219
 39024 :073,188,032,221,189,162,209
 39030 :000,189,001,001,240,006,043
 39036 :157,119,002,232,208,245,063
 39042 :169,032,157,119,002,232,073
 39048 :134,198,076,131,164,032,103
 39054 :225,255,208,025,120,169,120
 39060 :131,141,002,003,169,164,246
 39066 :141,003,003,169,065,141,164
 39072 :020,003,169,147,141,021,149
 39078 :003,032,033,159,088,076,045
 39084 :065,147,169,195,141,008,129
 39090 :003,169,152,141,009,003,143
 39096 :169,000,141,030,144,141,041
 39102 :031,144,076,055,158,165,051
 39108 :058,201,255,240,039,173,138
 39114 :030,144,197,057,208,009,079
 39120 :173,031,144,197,058,208,251
 39126 :002,240,023,169,091,032,003
 39132 :210,255,166,057,142,030,056
 39138 :144,165,058,141,031,144,141
 39144 :032,201,189,169,093,032,180
 39150 :210,255,076,228,167,169,063
 39156 :228,141,008,003,169,167,192
 39162 :141,009,003,076,055,158,180
 39168 :032,157,155,169,010,141,152
 39174 :006,144,141,008,144,169,106
 39180 :000,141,011,144,141,007,200
 39186 :144,141,009,144,032,146,122
 39192 :145,176,017,140,006,144,140
 39198 :141,007,144,032,146,145,133
 39204 :176,006,140,008,144,141,139

39210 :009,144,160,000,177,188,208
 39216 :141,023,144,200,177,188,153
 39222 :013,023,144,240,058,200,220
 39228 :200,200,177,188,240,074,115
 39234 :201,044,208,005,173,011,196
 39240 :144,208,081,177,188,016,118
 39246 :238,201,137,240,073,201,144
 39252 :141,240,069,201,167,208,086
 39258 :019,140,023,144,200,177,025
 39264 :188,201,032,240,249,172,154
 39270 :023,144,201,058,144,048,208
 39276 :176,207,169,000,141,011,044
 39282 :144,240,200,169,001,141,241
 39288 :028,144,169,255,133,100,181
 39294 :133,101,032,051,165,032,128
 39300 :198,154,032,146,155,076,125
 39306 :049,158,200,152,024,101,054
 39312 :188,133,188,165,189,105,088
 39318 :000,133,189,076,044,153,233
 39324 :140,015,144,200,177,188,252
 39330 :201,032,240,249,172,015,047
 39336 :144,201,058,176,004,201,184
 39342 :048,176,003,076,061,153,179
 39348 :169,001,141,011,144,200,078
 39354 :024,152,101,188,133,122,138
 39360 :133,186,165,189,105,000,202
 39366 :133,123,133,187,140,023,169
 39372 :144,032,073,155,169,000,009
 39378 :141,028,144,165,122,133,175
 39384 :146,165,123,133,147,032,194
 39390 :051,165,032,198,154,032,086
 39396 :084,155,024,101,186,133,143
 39402 :148,165,187,105,000,133,204
 39408 :149,169,004,141,029,144,108
 39414 :032,013,154,172,023,144,016
 39420 :162,255,232,189,000,001,067
 39426 :208,003,076,062,153,145,137
 39432 :188,200,076,254,153,032,143
 39438 :033,154,176,014,024,165,068
 39444 :182,109,029,144,133,182,031
 39450 :165,183,105,000,133,183,027
 39456 :096,056,165,148,229,146,104
 39462 :141,024,144,165,149,229,122
 39468 :147,013,024,144,208,001,069
 39474 :096,144,066,032,184,154,214
 39480 :165,181,170,005,180,208,197
 39486 :001,096,024,138,101,147,057
 39492 :141,099,154,165,146,141,146
 39498 :098,154,024,138,101,149,226
 39504 :141,102,154,165,148,141,163
 39510 :101,154,232,164,180,208,101
 39516 :004,240,013,160,255,185,181
 39522 :000,000,153,000,000,136,131
 39528 :192,255,208,245,206,099,029
 39534 :154,206,102,154,202,208,112
 39540 :234,024,096,032,184,154,072
 39546 :165,146,141,154,154,165,023
 39552 :147,141,155,154,165,148,014
 39558 :141,157,154,165,149,141,017

39564 :158,154,166,181,240,032,047
 39570 :169,000,141,010,144,160,002
 39576 :000,185,000,000,153,000,234
 39582 :000,200,204,010,144,208,156
 39588 :244,238,155,154,238,158,071
 39594 :154,224,000,240,007,202,229
 39600 :208,224,165,180,208,222,103
 39606 :056,096,056,165,182,229,198
 39612 :146,133,180,165,183,229,200
 39618 :147,133,181,096,173,006,162
 39624 :144,133,184,173,007,144,217
 39630 :133,185,173,004,144,133,210
 39636 :253,173,005,144,133,254,150
 39642 :160,000,177,253,133,167,084
 39648 :200,177,253,133,168,005,136
 39654 :167,240,075,200,177,253,062
 39660 :056,229,101,141,013,144,152
 39666 :200,177,253,229,100,013,190
 39672 :013,144,208,018,173,028,064
 39678 :144,208,012,169,032,032,083
 39684 :210,255,166,184,165,185,145
 39690 :032,205,189,096,176,036,232
 39696 :173,028,144,240,015,152,000
 39702 :072,160,002,165,184,145,238
 39708 :253,200,165,185,145,253,205
 39714 :104,168,165,184,024,109,020
 39720 :008,144,133,184,165,185,091
 39726 :109,009,144,076,060,155,087
 39732 :169,255,133,184,133,185,087
 39738 :208,192,133,185,165,167,084
 39744 :133,253,165,168,133,254,146
 39750 :076,218,154,032,158,173,113
 39756 :032,155,188,165,100,164,112
 39762 :101,096,160,255,200,185,055
 39768 :000,001,208,250,152,096,027
 39774 :165,043,133,251,165,044,127
 39780 :133,252,160,000,177,251,049
 39786 :208,018,232,224,003,208,231
 39792 :015,152,056,101,251,133,052
 39798 :045,165,252,105,000,133,050
 39804 :046,096,162,000,200,208,068
 39810 :229,230,252,208,225,162,156
 39816 :044,181,146,157,211,159,010
 39822 :202,016,248,096,162,044,142
 39828 :189,211,159,149,146,202,180
 39834 :016,248,096,032,135,155,068
 39840 :165,045,024,105,005,133,125
 39846 :182,165,046,105,000,133,029
 39852 :183,165,043,133,188,141,001
 39858 :004,144,165,044,133,189,089
 39864 :141,005,144,096,169,000,227
 39870 :141,011,144,169,001,141,029
 39876 :013,144,208,020,169,001,239
 39882 :141,011,144,169,000,141,040
 39888 :013,144,240,008,169,000,014
 39894 :141,011,144,141,013,144,040
 39900 :032,124,165,032,026,157,244
 39906 :032,157,155,169,255,141,111
 39912 :008,144,141,009,144,169,079

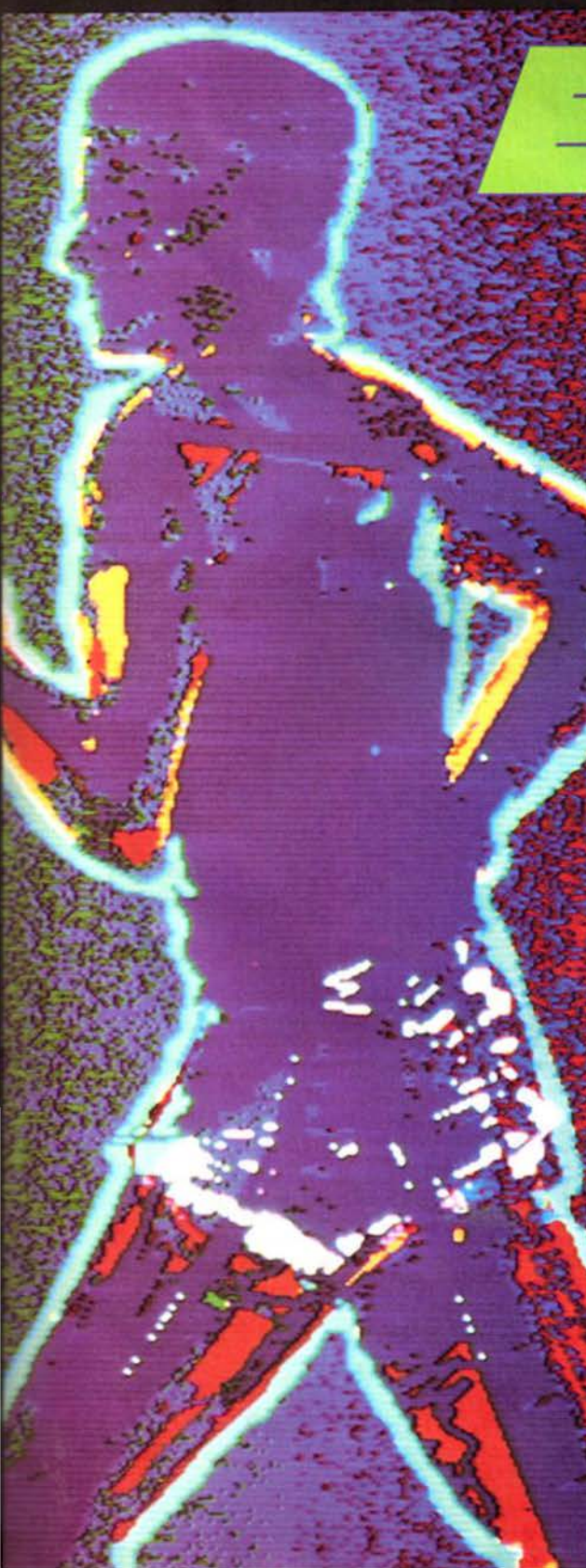


39918 :002,133,123,032,146,145,051
 39924 :176,026,132,020,133,021,240
 39930 :032,019,166,165,095,133,092
 39936 :188,165,096,133,189,032,035
 39942 :146,145,176,006,140,008,115
 39948 :144,141,009,144,160,000,098
 39954 :177,188,141,023,144,200,123
 39960 :177,188,013,023,144,240,041
 39966 :072,162,000,200,177,188,061
 39972 :133,020,237,008,144,200,010
 39978 :177,188,133,021,237,009,039
 39984 :144,144,002,176,050,200,252
 39990 :177,188,240,029,205,011,136
 39996 :001,208,246,140,023,144,054
 40002 :162,000,200,232,189,011,092
 40008 :001,201,064,240,035,209,054
 40014 :188,240,243,172,023,144,064
 40020 :076,053,156,200,152,024,233
 40026 :101,188,133,188,165,189,030
 40032 :105,000,133,189,076,016,103
 40038 :156,032,051,165,032,146,172
 40044 :155,076,049,158,140,024,198
 40050 :144,032,051,165,172,023,189
 40056 :144,136,192,004,176,001,005
 40062 :200,132,180,177,188,141,120
 40068 :003,144,174,013,144,240,082
 40074 :004,169,113,145,188,032,021
 40080 :019,166,169,159,141,000,030
 40086 :003,169,156,141,001,003,111
 40092 :076,201,166,169,139,141,024
 40098 :000,003,169,227,141,001,191



40104	:003,173,003,144,164,180,067	40458	:146,048,006,032,210,255,195
40110	:145,188,172,024,144,173,252	40464	:076,000,158,073,128,032,227
40116	:011,144,240,003,076,054,196	40470	:210,255,165,251,073,001,209
40122	:156,174,013,144,240,022,167	40476	:133,251,208,009,169,000,030
40128	:165,197,197,203,240,252,166	40482	:133,211,032,033,159,208,042
40134	:032,060,145,165,203,201,236	40488	:004,169,020,133,211,200,009
40140	:025,240,007,201,039,208,156	40494	:200,208,207,032,094,155,174
40146	:243,076,054,156,152,024,147	40500	:032,092,166,169,000,133,132
40152	:101,188,133,146,165,189,114	40506	:198,162,128,108,000,003,145
40158	:105,000,133,147,173,023,035	40512	:169,000,133,247,169,206,220
40164	:144,024,109,029,144,133,043	40518	:133,248,169,000,133,249,234
40170	:148,032,251,156,165,148,110	40524	:169,207,133,250,032,231,074
40176	:024,101,188,133,148,032,098	40530	:255,169,001,162,032,160,093
40182	:251,156,076,002,157,165,029	40536	:159,032,189,255,169,002,126
40188	:189,105,000,133,149,096,156	40542	:170,160,003,032,186,255,132
40194	:032,013,154,172,023,144,028	40548	:032,192,255,032,197,158,198
40200	:162,255,232,189,041,001,120	40554	:032,204,255,032,228,255,088
40206	:201,064,208,003,076,054,108	40560	:201,000,240,031,201,092,109
40212	:156,145,188,200,208,240,133	40566	:240,066,168,120,162,054,160
40218	:160,000,200,185,000,002,061	40572	:134,001,185,000,162,162,000
40224	:240,056,201,064,208,246,023	40578	:055,134,001,088,072,162,130
40230	:162,255,232,200,185,000,048	40584	:002,032,201,255,104,032,250
40236	:002,240,043,201,034,240,036	40590	:210,255,032,204,255,162,236
40242	:246,157,011,001,201,064,218	40596	:002,032,198,255,032,228,127
40248	:208,238,173,011,144,208,014	40602	:255,201,000,240,203,041,070
40254	:023,162,255,232,200,185,095	40608	:127,168,120,162,054,134,157
40260	:000,002,240,018,201,034,051	40614	:001,185,000,160,162,055,217
40266	:240,246,157,041,001,201,192	40620	:134,001,088,072,032,204,191
40272	:064,208,238,142,029,144,137	40626	:255,104,032,210,255,076,086
40278	:200,132,122,096,162,011,041	40632	:106,158,032,204,255,169,084
40284	:108,000,003,120,072,138,021	40638	:002,032,195,255,076,055,037
40290	:072,152,072,169,004,141,196	40644	:158,160,255,152,192,219,052
40296	:136,002,032,188,246,032,228	40650	:176,024,192,193,144,006,169
40302	:225,255,240,003,076,114,255	40656	:056,233,128,076,230,158,065
40308	:254,032,163,253,032,024,106	40662	:192,065,144,012,192,096,147
40314	:229,173,021,144,141,033,095	40668	:176,006,024,105,032,076,127
40320	:208,173,022,144,141,032,080	40674	:230,158,169,000,153,000,168
40326	:208,173,020,144,141,134,186	40680	:162,136,192,255,208,217,122
40332	:002,108,002,160,169,004,073	40686	:152,192,128,176,024,192,078
40338	:162,004,160,004,032,186,182	40692	:096,144,006,056,233,032,043
40344	:255,032,192,255,162,004,028	40698	:076,013,159,192,065,144,131
40350	:032,201,255,160,255,200,237	40704	:012,192,091,176,008,024,247
40356	:192,087,240,014,185,000,114	40710	:105,128,076,013,159,169,144
40362	:002,201,045,208,244,169,015	40716	:000,153,000,160,136,192,141
40368	:171,153,000,002,208,237,179	40722	:255,208,217,169,008,141,248
40374	:169,198,141,000,003,169,094	40728	:020,162,169,020,141,008,032
40380	:157,141,001,003,032,115,125	40734	:160,096,006,169,013,032,250
40386	:000,076,156,166,169,139,132	40740	:210,255,096,032,146,145,152
40392	:141,000,003,169,227,141,113	40746	:032,189,255,032,146,145,073
40398	:001,003,162,004,032,201,097	40752	:132,251,133,252,032,146,226
40404	:255,169,013,032,210,255,122	40758	:145,169,002,162,008,168,196
40410	:032,204,255,169,002,032,144	40764	:032,186,255,032,192,255,244
40416	:195,255,076,055,158,032,227	40770	:032,080,145,176,009,169,165
40422	:146,145,140,012,144,076,125	40776	:251,166,101,164,100,032,118
40428	:055,158,169,000,133,251,234	40782	:216,255,169,002,032,195,179
40434	:169,147,032,210,255,169,200	40788	:255,076,055,158,169,001,030
40440	:004,133,214,032,033,159,055	40794	:133,212,165,045,133,251,005
40446	:160,255,200,192,219,144,144	40800	:096,072,013,013,013,255,046
40452	:003,076,055,158,185,081,050		

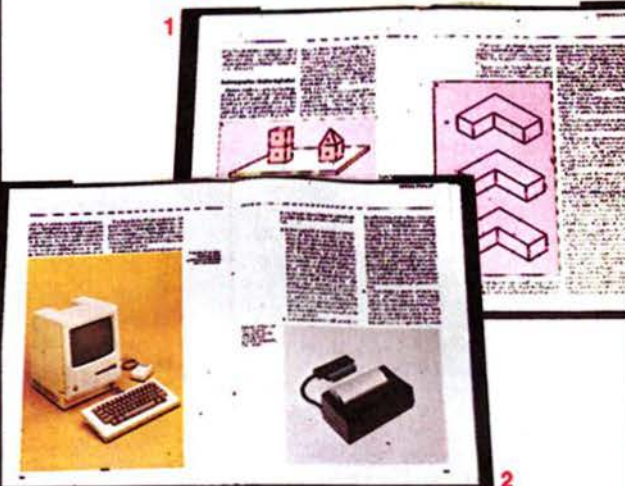
NON FARTI SUPERARE DAL PROGRESSO



Aggiornati con gli "Aggiornamenti"

ENCICLOPEDIA DI ELETTRONICA & INFORMATICA

20 FASCICOLI SETTIMANALI DA RILEGARE
IN DUE NUOVI E SPENDIDI VOLUMI



PER TE, PER IL TUO LAVORO, PER I TUOI STUDI...

1 **Aggiornamenti**

le nuove conquiste dell'Elettronica di Base, delle Comunicazioni, dell'Elettronica Digitale, dei Microprocessori, dell'Informatica...

2 **Il personal computer**

tutto quello che c'è da sapere sul Personal Computer: che cos'è e cosa fa; come fa e come si fa; i linguaggi di programmazione; le applicazioni...

E.I. si aggiorna e ti aggiorna

il 18 settembre

batti sul tempo il progresso!
Corri in edicola a comprare il
primo fascicolo.

Solo L. 2500

**ANCORA UNA VOLTA
"PRIMI SUL FUTURO"**



**GRUPPO
EDITORIALE
JACKSON**

SAN FRANCISCO - LONDRA - MILANO



Pool

di J. T. Woyton
trad. ed. adatt.
di M. Cristulb Grizzi

Prendete in mano la stecca e mostrate la vostra abilità in questa entusiasmante simulazione del biliardo. Per due giocatori, in versione per VIC 20 inespanso e C64.

Le regole di POOL sono molto semplici: si cerca di far cadere in buca tutte le palle sul tavolo, tirando in modo appropriato la palla bianca.

La versione per VIC 20 accetta i comandi sia da tastiera che da joystick (le differenze nella versione per C64 sono spiegate alla fine dell'articolo). Usate il joystick (o il tasto F3) per ruotare la stecca intorno alla palla da tirare.

Il tiro viene effettuato premendo il pulsante di fuoco (oppure il tasto F1).

Più a lungo si tiene premuto il pulsante di fuoco (od il tasto F1), meno forte sarà il tiro nel momento in cui si rilascerà il pulsante stesso. Per un tiro forte premete il pulsante di fuoco (o F1) e rilasciatelo immediatamente.

Il vostro turno continua fino a che riuscite a mandare le palle in buca. Se sbagliate o mandate in buca la palla bianca, il turno passa all'avversario. Nel caso mandate in buca la pallina bianca, essa potrà essere riposizionata con il joystick (oppure i tasti F3 ed F5) in qualunque luogo al di sotto della linea che verrà visualizzata sullo schermo. Premete il pulsante di fuoco quando la palla si trova nella posizione desiderata.

Le quindici palle vengono posizionate a caso sul tavolo dopo che l'ultima è finita in buca: potrete a questo punto scegliere se continuare il gioco (fino ad un limite stabilito tra voi ed il vostro avversario), oppure iniziare una nuova partita.

Note alla versione per VIC 20

La routine principale di gioco fa in modo che le palle sul tavolo da biliardo si muovano, rimbalzino ed entrino in collisione tra di loro. In questa routine il moto è simulato "pokando" il carattere che rappresenta la palla nella locazione di schermo successiva, a seconda della



direzione del moto, e quindi "pokando" uno spazio nella posizione precedente, cancellando quindi il carattere. La risoluzione dello schermo del VIC 20 permette soltanto otto direzioni di movimento.

Il programma esegue delle PEEK nelle locazioni antecedenti la posizione corrente della palla, per rilevare eventuali collisioni. Quando viene raggiunto il bordo del tavolo il verso del moto viene invertito. Se vengono rilevate le buche (CHR\$ (102)), viene chiamata la subroutine di punteggio.

Il rallentamento della palla durante il suo moto è ottenuto con cicli di ritardo

progressivamente più lunghi tra una POKE e l'altra nelle locazioni della memoria di schermo. Effetti sonori sottolineano le collisioni e l'incremento del punteggio.

Dal momento che il programma occupa quasi totalmente la memoria disponibile nel VIC 20, si raccomanda a chi non disponesse della cassetta allegata alla rivista di non inserire nel listato spazi superflui od istruzioni REM aggiuntive.

Note alla versione per C64

Il programma per C64 è interamente in linguaggio macchina ed utilizza l'alta risoluzione grafica. Richiede inoltre l'uti-

lizzo di uno o due joystick (a seconda del numero di giocatori). Se si dispone della cassetta allegata alla rivista, è sufficiente posizionarsi con il nastro in corrispondenza dell'inizio del programma e digitare:

LOAD"" ,1,1

(attenzione: non inserire spazi tra gli apici!).

Il programma andrà quindi mandato in esecuzione, digitando:

SYS 49152

Ricordiamo che per conoscere il punto esatto di inizio dei vari programmi su cassetta è sufficiente inserire quest'ultima, completamente riavvolta, nel registratore, azzerare il contanastro e digitare:

LOAD""

Dal momento che sicuramente non esiste sulla cassetta alcun programma di nome "", il computer passerà in rassegna tutti i programmi incontrati, visualizzando il nome con il messaggio:

FOUND XXXX

Sarà quindi utile annotarsi su un foglio i nomi dei programmi e la relativa posizione del contanastro (sottrarre tre o

quattro unità dal valore corrispondente all'apparizione del messaggio FOUND ...).

Gli abbonati alla sola rivista dovranno invece usare il programma MLX (vedere nelle pagine verdi) per digitare il listato in linguaggio macchina. La procedura da seguire, in questo caso, è: caricare in memoria MLX (che sarà stato precedentemente digitato e salvato) e rispondere alle sue richieste circa gli indirizzi iniziale e finale con:

Indirizzo iniziale: 49152

Indirizzo finale: 52905

Dopo avere digitato e salvato il programma, tramite MLX, si potranno seguire per il caricamento le istruzioni date più sopra per i possessori di cassetta. Mandando in esecuzione il programma, comparirà il titolo iniziale, con una melodia rag-time in sottofondo. Verrà quindi richiesto di scegliere tra uno o due giocatori, quindi apparirà il tavolo da biliardo. Nella versione per C64 ci sono solo sei palle, oltre la palla bianca: ciò per la nota limitazione sul massimo numero di SPRITE visualizzabili contemporaneamente sullo schermo (otto).

Con il joystick posizionate la palla bianca al di sotto della linea visualizzata e, quando siete pronti per tirare, premete il pulsante di fuoco. Apparirà quindi una croce, che può essere spostata con il joystick nella direzione in cui desiderate tirare. Al contrario della versione per

VIC 20, che permette soltanto otto direzioni, potete qui scegliere una direzione qualsiasi. Premendo il pulsante di fuoco inizierà il movimento della palla nella direzione voluta. La distanza tra palla e croce misura la forza del tiro: più grande è questa distanza, più forte sarà il tiro. Esiste comunque un limite su questa distanza, pari a circa due terzi della lunghezza del tavolo. Nel gioco singolo l'obiettivo è quello di mandare in buca tutte le palle con il minor numero possibile di tiri; il computer stesso offrirà una valutazione della vostra abilità.

Nel gioco contro un avversario, quando si manda in buca una palla questa viene visualizzata nella propria metà di schermo. Si perde un tiro quando si getta in buca la palla che si è lanciata, o quando quest'ultima non tocca nulla prima di fermarsi: il turno viene perduto, una delle palle precedentemente guadagnate e visualizzate nella propria metà di schermo viene rimessa in gioco e la palla tirata viene riportata nella posizione di partenza. Toccherà a questo punto all'avversario di posizionare la palla nel punto desiderato ed effettuare il proprio tiro.

Il programma non segue completamente le leggi della fisica nel moto delle palle, pur offrendo una simulazione molto realistica. Gli SPRITE sono mossi pixel per pixel, ma il movimento è calcolato in duecentocinquantesimesimi di pixel, per avere la massima accuratezza nei calcoli delle posizioni.

Pool versione per VIC 20

```
100 POKE36879,10:PRINT" {CLR} {YEL}" :rem 154
110 PRINT"NOME PRIMO GIOCATORE" :rem 182
120 PRINT:INPUTN$ :rem 92
130 A$=LEFT$(N$,7) :rem 171
140 PRINT:PRINT:PRINT"NOME SECONDO GIOCATORE" :rem 203
150 PRINT:INPUTN$ :rem 95
160 B$=LEFT$(N$,7) :rem 175
170 PRINT" {CLR}" :rem 252
180 D=1:Q0=7981:B0=87:B1=81:B=B0:C=1 :rem 221
190 POKE37154,127:POKE36878,15 :rem 158
200 FORP=38511TO38863STEP22 :rem 87
210 FORZ=0TO19 :rem 78
220 POKEP+Z,7 :rem 251
230 NEXTZ:NEXTP :rem 247
240 B=B0:Q=Q0 :rem 208
250 POKEQ,B :rem 133
260 GOSUB960 :rem 183
270 REM GIOCO :rem 238
280 IFD=1THENGOSUB430 :rem 33
290 GOSUB820 :rem 181
300 IFPEEK(Q+I)<>32THENGOSUB730 :rem 133
310 IFPEEK(Q+I)<>32THENGOSUB820 :rem 134
320 IFPEEK(Q+I)<>32THENGOSUB730 :rem 135
330 POKEQ,32 :rem 167
340 Q=Q+I :rem 234
350 IFB=B0THENQC=Q :rem 105
360 POKEQ,B :rem 135
370 IFI=0THEN280 :rem 168
```

```
380 FORT=0TOD:NEXTT :rem 247
390 D=D*1.2 :rem 28
400 IFD>200THENI=0:D=1:H=0:IFB=B1THENB=B0 :rem 86
:Q=QC:POKEQ,B :rem 33
410 IFD=1THENGOSUB930 :rem 105
420 GOTO280 :rem 123
430 REM :rem 124
440 REM :rem 101
450 IFS=1THENGOSUB1570 :rem 32
460 FORV=1TO8 :rem 63
470 IFV=1THENI=-1:Z=67 :rem 118
480 IFV=2THENI=-23:Z=77 :rem 117
490 IFV=3THENI=-22:Z=93 :rem 112
500 IFV=4THENI=-21:Z=78 :rem 17
510 IFV=5THENI=1:Z=67 :rem 72
520 IFV=6THENI=23:Z=77 :rem 71
530 IFV=7THENI=22:Z=93 :rem 75
540 IFV=8THENI=21:Z=78 :rem 85
550 A=PEEK(Q-I) :rem 23
560 POKEQ-I,Z :rem 69
570 FORT=0TO100:NEXTT :rem 0
580 POKEQ-I,A :rem 233
590 GOSUB1820 :rem 106
600 IFJ=0ANDK=64THEN550 :rem 112
610 IFJ=32ORK=39THEN640 :rem 45
620 NEXTV :rem 105
630 GOTO430 :rem 95
640 REM VELOCITA' DELLA PALLINA :rem 5
650 FORD=1TO100STEP10 :rem 114
660 V$=STR$(100-(D-1)) :rem 64
670 PRINT" {HOME} { 2 GIU' }VELOC";V$+" " :rem 73
680 FORT=0TO300:NEXTT :rem 234
690 GOSUB1820
```



```

700 IFJ=0ANDK=64THEN1350 :rem 154
710 NEXTD :rem 27
720 GOTO640 :rem 108
730 REM CONFINI DEL CAMPO DI GIOCO :rem 199
740 IFPEEK(Q+I)=102THENH=1:GOTO910 :rem 39
750 IFPEEK(Q+22)=102THENH=1:GOTO910 :rem 67
760 IFPEEK(Q-22)=102THENH=1:GOTO910 :rem 70
770 IFPEEK(Q+I)=100THENI=I+44:GOTO1350 :rem 3
780 IFPEEK(Q+I)=99THENI=I-44:GOTO1350 :rem 231
790 IFPEEK(Q+I)=103THENI=I+2:GOTO1350 :rem 210
800 IFPEEK(Q+I)=101THENI=I-2:GOTO1350 :rem 202
810 RETURN :rem 121
820 REM COLLISIONE TRA LE PALLINE :rem 236
830 IFI=0THENRETURN :rem 239
840 IFPEEK(Q+I)=B0THENPOKEQ,B:B=B0:Q=Q+I: :rem 38
GOTO1350 :rem 41
850 IFPEEK(Q+I)=B1THENPOKEQ,B:B=B1:Q=Q+I: :rem 250
GOTO1350 :rem 102
860 IFPEEK(Q+1)=B1THENPOKEQ,B:B=B1:Q=Q+1: :rem 255
GOTO1350 :rem 102
870 IFPEEK(Q-1)=B1THENPOKEQ,B:B=B1:Q=Q-1: :rem 99
GOTO1350 :rem 121
880 IFPEEK(Q-22)=B1THENPOKEQ,B:B=B1:Q=Q-2 :rem 99
2:GOTO1350 :rem 106
890 IFPEEK(Q+22)=B1THENPOKEQ,B:B=B1:Q=Q+2 :rem 127
2:GOTO1350 :rem 237
900 RETURN :rem 158
910 REM GESTIONE DEL PUNTEGGIO :rem 154
920 IFB=B0THENI=0:D=1:POKEQ,32:Q=Q0:H=0:S :rem 104
=1 :rem 177
930 IFH=0THENC=-C:GOSUB1500 :rem 105
940 IFC<0THEN1040 :rem 213
950 IFH=1THENSA=SA+1:POKEQ,32:GOSUB1410 :rem 81
:rem 12
1000 IFPEEK(P)=B1THENRETURN :rem 164
1010 NEXTP :rem 199
1020 GOSUB1080 :rem 11
1030 RETURN :rem 66
1040 IFH=1THENSB=SB+1:POKEQ,32:GOSUB1410 :rem 75
:rem 161
1050 PRINT"(HOME){RVS}"A$;SA;TAB(11)"{RVS} :rem 173
{CYN}"B$;SB"{OFF}{YEL}" :rem 66
1060 IFSA+SB=15THEN1860 :rem 75
1070 GOTO970 :rem 161
1080 REM :rem 173
1090 FORP=7770TO7777:POKEP,100:NEXT :rem 129
:rem 123
1100 FORP=7780TO7787:POKEP,100:NEXT :rem 82
:rem 85
1110 FORP=8166TO8173:POKEP,99:NEXT :rem 11
:rem 11
1120 FORP=8176TO8183:POKEP,99:NEXT :rem 11
:rem 11
1130 FORP=7812TO8120STEP22:POKEP,103:NEXT :rem 11
:rem 11

```

```

1140 FORP=7833TO8141STEP22:POKEP,101:NEXT :rem 16
:rem 16
1150 POKE7768,102:POKE7769,102:POKE7790,1 :rem 149
02 :rem 124
1160 POKE8164,102:POKE8165,102:POKE8142,1 :rem 149
02 :rem 133
1170 POKE7789,102:POKE7788,102:POKE7811,1 :rem 156
02 :rem 130
1180 POKE8185,102:POKE8184,102:POKE8163,1 :rem 112
02 :rem 198
1190 POKE7778,102:POKE7779,102 :rem 238
1200 POKE8174,102:POKE8175,102 :rem 75
1210 FORN=1TO15 :rem 78
1220 P=7768+INT(396*RND(0)) :rem 128
1230 IFPEEK(P)<>32THEN1220 :rem 131
1240 IFPEEK(P+1)<>32THEN1220 :rem 129
1250 IFPEEK(P-1)<>32THEN1220 :rem 132
1260 IFPEEK(P+22)<>32THEN1220 :rem 124
1270 IFPEEK(P-22)<>32THEN1220 :rem 127
1280 IFPEEK(P+21)<>32THEN1220 :rem 228
1290 IFPEEK(P-21)<>32THEN1220 :rem 84
1300 IFPEEK(P+23)<>32THEN1220 :rem 168
1310 IFPEEK(P-23)<>32THEN1220 :rem 17
1320 POKEP,B1 :rem 74
1330 NEXTN :rem 204
1340 RETURN :rem 69
1350 REM SUONO DELLA COLLISIONE TRA LE PA :rem 101
LLINE :rem 165
1360 D=D*1.2 :rem 155
1370 POKE36875,180 :rem 67
1380 FORT=0TO10:NEXTT :rem 197
1390 POKE36875,0 :rem 67
1400 RETURN :rem 99
1410 REM SUONO DELL' ACCREDITO DEL PUNTEG :rem 67
GIO :rem 174
1420 FORS=0TO2 :rem 170
1430 POKE36876,220 :rem 200
1440 FORT=0TO20:NEXTT :rem 194
1450 POKE36876,0 :rem 118
1460 FORT=0TO20:NEXTT :rem 98
1470 IFSA+SB=15THEN1860 :rem 98
1480 NEXTS :rem 172
1490 RETURN :rem 177
1500 REM :rem 178
1510 POKE36875,135 :rem 50
1520 POKE36874,201 :rem 44
:rem 87
1530 FORT=0TO500:NEXTT :rem 173
1540 POKE36875,0 :rem 142
1550 POKE36874,0 :rem 22
1560 RETURN :rem 209
1570 REM :rem 209
1580 REM :rem 164
1590 FORP=7782TO8156STEP22 :rem 157
1600 IFPEEK(P)=32THENPOKEP,103 :rem 243
1610 NEXTP :rem 218
1620 REM :rem 194
1630 V=1 :rem 183
1640 GOSUB1820 :rem 206
1650 IFJ=0ANDK=64THEN1640 :rem 22
1660 IFJ=4ORJ=8ORJ=47THENI=V*22 :rem 212
1670 IFJ=1ORJ=16ORJ=55THENI=V*1 :rem 209
1680 IFPEEK(Q+I)<>32THENV=-V:I=-I :rem 178
:rem 50
1690 IFJ=32ORJ=39THEN1730 :rem 212
1700 POKEQ,32:Q=Q+I:QC=Q :rem 194
1710 POKEQ,B :rem 183
1720 GOTO1640 :rem 206
1730 GOSUB1820 :rem 22
1740 IFJ=0ANDK=64THEN1760 :rem 212
1750 GOTO1730 :rem 209
1760 REM :rem 178
1770 FORP=7782TO8156STEP22 :rem 50

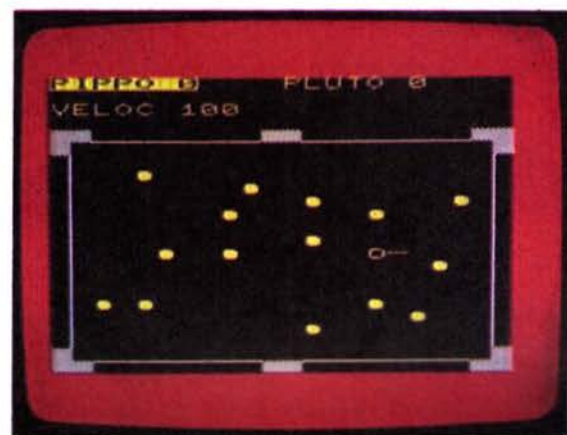
```



```

1780 IFPEEK(P)=103THENPOKEP,32 :rem 53
1790 NEXTP :rem 96
1800 S=0 :rem 137
1810 RETURN :rem 170
1820 REM LEGGE IL JOYSTICK :rem 24
1830 J=(NOTPEEK(37151))AND60-((PEEK(37152)
AND128)=0) :rem 103
1840 K=PEEK(197) :rem 108
1850 RETURN :rem 174
1860 PRINT"{CLR}{ 5 GIU' }{ 9 SPAZI }ANCORA
?":POKE198,0:PRINT"{ 2 GIU' }
{ 8 SPAZI }(S/N)" :rem 48
1870 GETAS:IFA$="S"THENRUN :rem 59
1880 IFA$<>"N"THEN1870 :rem 208

```



Pool versione per C64

```

49152 :169,012,141,021,208,169,208
49158 :255,141,059,099,141,060,249
49164 :099,169,001,141,001,088,255
49170 :169,000,141,051,099,032,254
49176 :024,197,032,206,203,169,087
49182 :255,141,021,208,032,209,128
49188 :195,032,093,197,032,163,236
49194 :197,032,248,204,169,000,124
49200 :141,062,099,141,063,099,141
49206 :173,001,088,208,115,162,033
49212 :000,142,057,099,189,004,039
49218 :208,201,070,240,006,201,224
49224 :235,240,008,208,097,238,074
49230 :062,099,076,086,192,238,063
49236 :063,099,232,232,224,012,178
49242 :208,228,162,022,160,012,114
49248 :024,032,240,255,173,058,110
49254 :099,201,001,208,003,076,178
49260 :027,206,173,062,099,205,112
49266 :063,099,240,036,176,017,233
49272 :169,238,160,204,032,030,185
49278 :171,169,184,160,205,032,023

```

```

49284 :030,171,076,161,192,169,163
49290 :227,160,204,032,030,171,194
49296 :169,184,160,205,032,030,156
49302 :171,076,161,192,169,171,066
49308 :160,205,032,030,171,169,155
49314 :060,133,162,165,161,197,016
49320 :161,240,252,076,000,192,065
49326 :173,001,088,240,003,032,199
49332 :010,195,169,000,141,001,184
49338 :088,141,061,099,032,226,065
49344 :192,169,000,141,164,096,186
49350 :141,032,096,141,000,088,184
49356 :032,042,193,162,009,032,162
49362 :112,193,202,208,250,032,183
49368 :118,193,173,000,088,240,004
49374 :226,076,043,192,173,002,166
49380 :208,141,000,208,201,175,137
49386 :144,004,169,254,208,003,248
49392 :024,105,080,141,053,099,230
49398 :173,000,208,056,233,080,228
49404 :141,054,099,173,003,208,162
49410 :141,001,208,201,165,144,094
49416 :004,169,245,208,003,024,149
49422 :105,080,141,056,099,173,156
49428 :057,099,208,017,173,001,063
49434 :208,201,130,176,004,169,146
49440 :050,208,003,056,233,080,150
49446 :141,055,099,096,174,000,091
49452 :208,172,051,099,185,000,247
49458 :220,172,001,208,074,176,133
49464 :007,136,204,055,099,176,221
49470 :001,200,074,176,007,200,208
49476 :204,056,099,144,001,136,196
49482 :074,176,007,202,236,054,055
49488 :099,176,001,232,074,176,070
49494 :007,232,236,053,099,144,089
49500 :001,202,074,176,008,169,210
49506 :001,141,032,096,141,000,253
49512 :088,142,000,208,140,001,171
49518 :208,096,160,000,136,208,150
49524 :253,096,173,032,096,208,206
49530 :003,076,143,194,169,000,195
49536 :141,112,096,141,096,096,042
49542 :173,000,208,056,237,002,042
49548 :208,046,112,096,141,000,231
49554 :096,173,001,208,056,237,149
49560 :003,208,046,096,096,141,230
49566 :016,096,173,112,096,208,091
49572 :011,173,000,096,073,255,004
49578 :141,000,096,238,000,096,229
49584 :169,004,141,128,096,173,119
49590 :096,096,208,011,173,016,014
49596 :096,073,255,141,016,096,097
49602 :238,016,096,160,000,185,121
49608 :112,096,240,028,185,000,093
49614 :101,024,121,000,096,153,189
49620 :000,101,185,002,208,105,045
49626 :000,201,198,144,003,032,028
49632 :144,194,153,002,208,076,233

```


49638 :001,194,185,000,101,056,255
 49644 :249,000,096,153,000,101,067
 49650 :185,002,208,233,000,201,047
 49656 :106,176,003,032,144,194,135
 49662 :153,002,208,185,096,096,226
 49668 :240,028,185,032,101,024,102
 49674 :121,016,096,153,032,101,017
 49680 :185,003,208,105,000,201,206
 49686 :194,144,003,032,181,194,002
 49692 :153,003,208,076,059,194,209
 49698 :185,032,101,056,249,016,161
 49704 :096,153,032,101,185,003,098
 49710 :208,233,000,201,090,176,186
 49716 :003,032,181,194,153,003,106
 49722 :208,032,199,197,172,162,004
 49728 :096,200,200,192,014,240,238
 49734 :003,076,199,193,202,240,215
 49740 :003,076,197,193,206,128,111
 49746 :096,208,248,032,065,205,168
 49752 :169,004,141,128,096,162,020
 49758 :012,189,000,096,029,016,180
 49764 :096,201,006,176,228,169,208
 49770 :000,157,000,096,157,016,020
 49776 :096,202,202,016,234,169,007
 49782 :000,141,032,096,162,014,051
 49788 :157,000,096,157,016,096,134
 49794 :202,016,247,173,061,099,160
 49800 :208,005,169,001,141,001,149
 49806 :088,096,072,032,190,205,057
 49812 :162,001,185,112,096,073,009
 49818 :001,153,112,096,185,003,192
 49824 :208,201,192,176,043,201,157
 49830 :093,144,039,201,138,176,189
 49836 :002,104,096,201,146,144,097
 49842 :029,104,096,072,032,190,189
 49848 :205,162,000,185,096,096,160
 49854 :073,001,153,096,096,185,026
 49860 :002,208,201,195,176,006,216
 49866 :201,107,144,002,104,096,088
 49872 :032,208,205,104,192,000,181
 49878 :208,007,169,001,141,001,229
 49884 :088,208,003,238,164,096,249
 49890 :169,000,153,000,096,153,029
 49896 :016,096,173,051,099,240,139
 49902 :004,169,070,208,002,169,092
 49908 :235,153,002,208,152,010,236
 49914 :010,010,024,105,070,153,110
 49920 :003,208,224,000,240,003,166
 49926 :185,002,208,096,169,254,152
 49932 :141,021,208,173,058,099,200
 49938 :201,001,240,007,173,051,179
 49944 :099,073,001,240,004,169,098
 49950 :070,208,002,169,235,162,108
 49956 :000,221,004,208,240,009,206
 49962 :232,232,224,014,208,245,173
 49968 :076,096,195,169,155,157,128
 49974 :004,208,232,169,148,157,204
 49980 :004,208,254,004,208,165,135
 49986 :162,197,162,240,252,165,220

49992 :162,197,162,240,252,173,234
 49998 :030,208,208,236,254,004,250
 50004 :208,165,162,197,162,240,194
 50010 :252,173,030,208,208,222,159
 50016 :169,153,141,002,208,170,171
 50022 :169,100,141,003,208,141,096
 50028 :057,099,165,162,197,162,182
 50034 :240,252,172,051,099,185,089
 50040 :000,220,142,002,208,074,254
 50046 :176,017,206,003,208,072,040
 50052 :173,003,208,201,090,176,215
 50058 :005,169,100,141,003,208,252
 50064 :104,074,074,176,006,202,012
 50070 :224,105,176,001,232,074,194
 50076 :176,006,232,224,199,144,113
 50082 :001,202,074,176,199,165,211
 50088 :162,197,162,240,252,173,074
 50094 :030,208,041,002,208,186,081
 50100 :185,000,220,041,016,240,114
 50106 :249,169,000,133,162,105,236
 50112 :025,197,162,176,252,169,149
 50118 :255,141,021,208,173,003,231
 50124 :208,141,055,099,096,169,204
 50130 :147,032,210,255,169,160,159
 50136 :160,016,153,130,004,153,064
 50142 :170,004,169,160,153,002,112
 50148 :007,153,042,007,136,016,077
 50154 :239,032,076,204,169,210,140
 50160 :133,251,169,004,133,252,158
 50166 :169,210,133,253,169,216,116
 50172 :133,254,162,014,160,000,207
 50178 :169,160,145,251,200,169,072
 50184 :160,145,251,160,015,145,116
 50190 :251,200,169,160,145,251,166
 50196 :165,251,024,105,040,133,226
 50202 :251,165,252,105,000,133,164
 50208 :252,165,253,024,105,040,103
 50214 :133,253,165,254,105,000,180
 50220 :133,254,202,208,207,169,193
 50226 :000,141,051,099,169,033,031
 50232 :141,171,004,141,184,004,189
 50238 :141,195,005,141,219,006,001
 50244 :169,034,141,172,004,141,217
 50250 :185,004,141,209,005,141,247
 50256 :233,006,169,036,141,225,122
 50262 :004,141,249,005,141,017,131
 50268 :007,141,004,007,169,035,199
 50274 :141,211,004,141,235,005,067
 50280 :141,003,007,141,016,007,163
 50286 :169,174,141,175,004,141,146

LOAD

Pool

Gli ultimi saranno i primi?

Giudicale tu.

Anche se, in ordine di tempo, siamo gli ultimi a uscire con libri "dedicati" ai più diffusi home e personal computer, siamo convinti che il lettore attento ed esigente apprezzerà la qualità dei contenuti e la loro presentazione.

Abbiamo selezionato e tradotto i best-seller mondiali che accolgono quanto di più utile è stato scritto per il tuo computer.

Scegli a colpo sicuro!

Puoi ordinare direttamente i titoli presentati compilando ed inviando il coupon pubblicato oppure acquistarli presso i più qualificati computer shop e le migliori librerie.

J. soft: libri di qualità

Per Commodore 64

Giochi fantastici per il C 64
Cod. ASOC 001 L. 42.000
(inclusa cassetta)

Un libro pensato e realizzato per il divertimento dell'utente del C 64; contiene 19 giochi di vario genere tutti ampiamente commentati. L'attento studio dei programmi potrà essere vantaggioso per l'apprendimento delle tecniche di programmazione.

Il libro del C 64
Volume 1
Cod. ASOC 010 L. 24.000
(inclusa cassetta)
Il libro del C 64
Volume 2
Cod. ASOC 011 L. 24.000
(inclusa cassetta)

I due libri illustrano a fondo le possibilità del Commodore 64. Un compendio di utili consigli, e quant'altro è necessario per conoscere meglio il proprio calcolatore. Esempi pratici completano l'esposizione in modo chiaro ed esauriente.



C 64: suono e grafica - Volume 1
Cod. ASOC 008 L. 24.000
(inclusa cassetta)

C 64: suono e grafica - Volume 2
Cod. ASOC 009 L. 24.000
(inclusa cassetta)

Due volumi che non possono mancare nella biblioteca dell'utente Commodore. Una miniera di idee e suggerimenti per la programmazione del tuo personal computer. Una vasta serie di programmi esemplificativi guidano il lettore al miglior sfruttamento del Commodore 64.

Per ZX Spectrum

15 giochi grafici per ZX Spectrum
Cod. ASOC 002 L. 24.000
(inclusa cassetta)

Un fantastico volume con 15 entusiasmanti giochi grafici per il tuo ZX Spectrum. Tutti i programmi sono "pronti all'uso" e ampiamente commentati. I programmi sono scritti sfruttando appieno le capacità del calcolatore ed il loro studio può giovare a chiunque intenda affinare le proprie tecniche di programmazione.



Inoltre, i Paper Book: la raccolta dei programmi suddivisi per macchina di Paper Soft, il primo settimanale di software su carta per il tuo computer.

Paper Book - programmi per Apple II -
Cod. ASOC 005 L. 18.000
Paper Book - programmi per VIC 20 -
Cod. ASOC 003 L. 18.000
Paper Book - programmi per Commodore 64 -
Cod. ASOC 004 L. 18.000
Paper Book - programmi per Texas TI 99/4A -
Cod. ASOC 007 L. 18.000
Paper Book - programmi per ZX Spectrum -
Cod. ASOC 006 L. 18.000



In ogni volume, corredato di cassetta, oltre 40 programmi per il tuo computer: giochi grafici, utility, arcade, adventure una miscela esplosiva di software "pronto all'uso"!

J. soft s.r.l.

Viale Restelli, 5 - 20124 Milano - tel. 02/6880841-6880842-6880843

CEDOLA DI ORDINAZIONE - LIBRI

Compilare e spedire in busta chiusa a:
J. soft - Viale Restelli 5 - 20124 Milano
tel. 02/6880841-6880842-6880843

Ordino i seguenti libri per un importo totale di L.
+ L. 2.000 come contributo fisso per spese di spedizione

Cod. Cod.
Cod. Cod.
Cod. Cod.

- ☐ Contanti allegati
☐ Assegno allegato n°
☐ Ho spedito l'importo a mezzo vaglia postale
☐ Ho versato l'importo sul CCP n° 19445204 intestato a J. soft - Milano
☐ Pagherò in contrassegno al postino al ricevimento dei volumi

Nome

Cognome

Via

CAP Città Prov.

Se richiesta fattura - codice fiscale

Data

50292	:181,004,141,035,005,141,111	50640	:185,002,208,141,161,096,233
50298	:049,005,141,115,005,141,066	50646	:160,000,140,160,096,172,174
50304	:129,005,141,059,006,141,097	50652	:160,096,173,016,064,200,161
50310	:073,006,141,139,006,141,128	50658	:200,192,014,208,004,172,248
50316	:153,006,141,007,007,141,083	50664	:162,096,096,056,249,000,123
50322	:013,007,169,019,032,210,084	50670	:101,173,161,096,249,002,252
50328	:255,169,003,141,255,097,048	50676	:208,201,008,144,004,201,242
50334	:169,077,160,205,032,030,063	50682	:248,144,225,204,162,096,049
50340	:171,206,255,097,208,244,065	50688	:240,220,141,165,096,140,234
50346	:169,018,141,255,097,169,251	50694	:160,096,172,162,096,185,109
50352	:120,160,205,032,030,171,126	50700	:032,101,056,172,160,096,117
50358	:206,255,097,208,244,169,081	50706	:249,032,101,172,162,096,062
50364	:003,141,255,097,169,077,162	50712	:185,003,208,172,160,096,080
50370	:160,205,032,030,171,206,230	50718	:249,003,208,201,008,144,075
50376	:255,097,208,244,160,039,179	50724	:004,201,248,144,178,141,184
50382	:169,160,153,192,007,169,032	50730	:166,096,238,061,099,169,103
50388	:001,153,192,219,136,016,161	50736	:016,141,005,212,169,000,079
50394	:243,173,058,099,201,001,225	50742	:141,006,212,169,015,141,226
50400	:208,018,162,001,160,014,019	50748	:024,212,169,120,169,008,250
50406	:024,032,240,255,169,163,089	50754	:141,024,212,141,001,212,029
50412	:160,205,032,030,171,076,142	50760	:169,128,141,004,212,169,127
50418	:018,197,162,001,160,001,013	50766	:129,141,004,212,172,162,130
50424	:024,032,240,255,169,227,171	50772	:096,174,160,096,185,000,027
50430	:160,204,032,030,171,162,245	50778	:096,072,185,112,096,208,091
50436	:001,160,029,024,032,240,234	50784	:005,104,032,018,201,072,016
50442	:255,169,238,160,204,032,044	50790	:104,141,167,096,185,016,043
50448	:030,171,169,013,141,033,061	50796	:096,072,185,096,096,208,093
50454	:208,096,169,000,160,127,014	50802	:005,104,032,018,201,072,034
50460	:153,064,003,136,016,250,138	50808	:104,141,168,096,189,000,050
50466	:160,007,169,000,153,112,123	50814	:096,072,189,112,096,208,131
50472	:096,153,096,096,136,016,121	50820	:005,104,032,018,201,072,052
50478	:245,169,001,141,028,208,070	50826	:104,141,169,096,189,016,085
50484	:141,037,208,160,007,162,255	50832	:096,072,189,096,096,208,133
50490	:023,185,096,204,157,064,019	50838	:005,104,032,018,201,072,070
50496	:003,185,104,204,157,128,077	50844	:104,141,170,096,169,000,068
50502	:003,136,202,202,202,016,063	50850	:141,032,064,141,033,064,125
50508	:238,169,014,141,248,007,125	50856	:173,165,096,016,009,032,147
50514	:169,013,160,006,153,249,064	50862	:018,201,141,165,096,238,009
50520	:007,136,016,250,096,120,201	50868	:032,064,173,166,096,016,215
50526	:165,001,041,251,133,001,174	50874	:009,032,018,201,141,166,241
50532	:169,000,133,251,169,056,110	50880	:096,238,033,064,174,165,194
50538	:133,252,169,000,133,253,022	50886	:096,172,166,096,189,199,092
50544	:169,208,133,254,162,008,022	50892	:205,141,176,096,169,000,223
50550	:160,000,177,253,145,251,080	50898	:141,165,096,141,166,096,247
50556	:136,208,249,230,252,230,149	50904	:185,199,205,141,177,096,195
50562	:254,202,208,240,160,032,202	50910	:173,032,064,240,021,173,157
50568	:185,118,204,153,008,057,093	50916	:176,096,141,171,002,141,187
50574	:136,016,247,165,001,009,204	50922	:167,002,173,165,096,141,210
50580	:004,133,001,088,173,024,059	50928	:168,002,141,172,002,076,033
50586	:208,041,240,009,014,141,039	50934	:017,199,173,176,096,032,171
50592	:024,208,096,160,015,185,080	50940	:018,201,141,171,002,141,158
50598	:150,204,153,000,208,136,249	50946	:167,002,173,165,096,073,166
50604	:016,247,160,007,185,166,185	50952	:255,105,000,141,172,002,171
50610	:204,153,039,208,136,016,166	50958	:141,168,002,173,033,064,083
50616	:247,169,000,160,015,153,160	50964	:240,028,173,177,096,141,107
50622	:000,096,153,016,096,136,175	50970	:173,002,032,018,201,141,081
50628	:016,247,096,140,162,096,185	50976	:169,002,173,166,096,141,011
50634	:185,000,101,141,016,064,197	50982	:174,002,073,255,105,000,135
		50988	:141,170,002,076,075,199,195

50994 : 173,177,096,141,169,002,040
 51000 : 032,018,201,141,173,002,111
 51006 : 173,166,096,141,170,002,042
 51012 : 073,255,105,000,141,174,048
 51018 : 002,174,169,002,172,170,251
 51024 : 002,173,168,096,141,086,234
 51030 : 064,032,254,200,032,185,085
 51036 : 201,032,004,201,174,167,103
 51042 : 002,172,168,002,173,167,014
 51048 : 096,141,086,064,032,254,009
 51054 : 200,032,185,201,056,229,245
 51060 : 251,141,183,096,152,229,144
 51066 : 252,141,182,096,174,169,112
 51072 : 002,172,170,002,173,167,046
 51078 : 096,141,086,064,032,254,039
 51084 : 200,032,185,201,032,004,026
 51090 : 201,174,167,002,172,168,006
 51096 : 002,173,168,096,141,086,050
 51102 : 064,032,254,200,032,185,157
 51108 : 201,024,101,251,141,185,043
 51114 : 096,152,101,252,141,184,072
 51120 : 096,174,169,002,172,170,191
 51126 : 002,173,170,096,141,086,082
 51132 : 064,032,254,200,032,185,187
 51138 : 201,032,004,201,174,167,205
 51144 : 002,172,168,002,173,169,118
 51150 : 096,141,086,064,032,254,111
 51156 : 200,032,185,201,056,229,091
 51162 : 251,141,187,096,152,229,250
 51168 : 252,141,186,096,174,169,218
 51174 : 002,172,170,002,173,169,150
 51180 : 096,141,086,064,032,254,141
 51186 : 200,032,185,201,032,004,128
 51192 : 201,174,167,002,172,168,108
 51198 : 002,173,170,096,141,086,154
 51204 : 064,032,254,200,032,185,003
 51210 : 201,024,101,251,141,189,149
 51216 : 096,152,101,252,141,188,178
 51222 : 096,174,173,002,172,174,045
 51228 : 002,173,184,096,141,086,198
 51234 : 064,173,185,096,141,084,009
 51240 : 064,032,185,201,032,004,046
 51246 : 201,174,171,002,172,172,170
 51252 : 002,173,186,096,141,086,224
 51258 : 064,173,187,096,141,084,035
 51264 : 064,032,185,201,056,229,063
 51270 : 251,141,080,064,152,229,219
 51276 : 252,141,178,096,174,173,066
 51282 : 002,172,174,002,173,186,023
 51288 : 096,141,086,064,173,187,067
 51294 : 096,141,084,064,032,185,184
 51300 : 201,032,004,201,174,171,115
 51306 : 002,172,172,002,173,184,043
 51312 : 096,141,086,064,173,185,089
 51318 : 096,141,084,064,032,185,208
 51324 : 201,024,101,251,141,082,156
 51330 : 064,152,101,252,141,179,251
 51336 : 096,174,173,002,172,174,159
 51342 : 002,173,188,096,141,086,060

51348 : 064,173,189,096,141,084,127
 51354 : 064,032,185,201,032,004,160
 51360 : 201,174,171,002,172,172,028
 51366 : 002,173,182,096,141,086,078
 51372 : 064,173,183,096,141,084,145
 51378 : 064,032,185,201,056,229,177
 51384 : 251,141,081,064,152,229,078
 51390 : 252,141,180,096,174,173,182
 51396 : 002,172,174,002,173,182,133
 51402 : 096,141,086,064,173,183,177
 51408 : 096,141,084,064,032,185,042
 51414 : 201,032,004,201,174,171,229
 51420 : 002,172,172,002,173,188,161
 51426 : 096,141,086,064,173,189,207
 51432 : 096,141,084,064,032,185,066
 51438 : 201,024,101,251,141,083,015
 51444 : 064,152,101,252,141,181,111
 51450 : 096,076,024,201,169,000,048
 51456 : 141,084,064,096,133,251,001
 51462 : 132,252,096,173,190,096,177
 51468 : 073,001,141,190,096,096,097
 51474 : 073,255,024,105,001,096,060
 51480 : 174,162,096,172,160,096,116
 51486 : 169,001,153,112,096,157,206
 51492 : 112,096,157,096,096,153,234
 51498 : 096,096,172,162,096,173,069
 51504 : 178,096,016,010,032,018,142
 51510 : 201,072,169,000,153,112,249
 51516 : 096,104,153,000,096,173,170
 51522 : 179,096,016,010,032,018,161
 51528 : 201,072,169,000,153,096,251
 51534 : 096,104,153,016,096,172,203
 51540 : 160,096,173,180,096,016,037
 51546 : 010,032,018,201,072,169,080
 51552 : 000,153,112,096,104,153,202
 51558 : 000,096,173,181,096,016,152
 51564 : 010,032,018,201,072,169,098
 51570 : 000,153,096,096,104,153,204
 51576 : 016,096,173,032,064,240,229
 51582 : 015,174,162,096,222,002,029
 51588 : 208,174,160,096,254,002,002
 51594 : 208,076,154,201,174,162,089
 51600 : 096,254,002,208,174,160,014
 51606 : 096,222,002,208,173,033,116
 51612 : 064,240,013,174,162,096,137
 51618 : 222,003,208,174,160,096,001
 51624 : 254,003,208,096,174,162,041
 51630 : 096,254,003,208,174,160,045
 51636 : 096,222,003,208,096,142,179
 51642 : 085,064,140,087,064,169,027
 51648 : 000,141,190,096,173,086,110
 51654 : 064,016,022,173,084,064,109
 51660 : 032,018,201,141,084,064,232
 51666 : 173,086,064,073,255,105,198
 51672 : 000,141,086,064,032,009,036
 51678 : 201,152,016,018,138,032,011
 51684 : 018,201,141,085,064,152,121
 51690 : 073,255,105,000,141,087,127
 51696 : 064,032,009,201,169,000,203

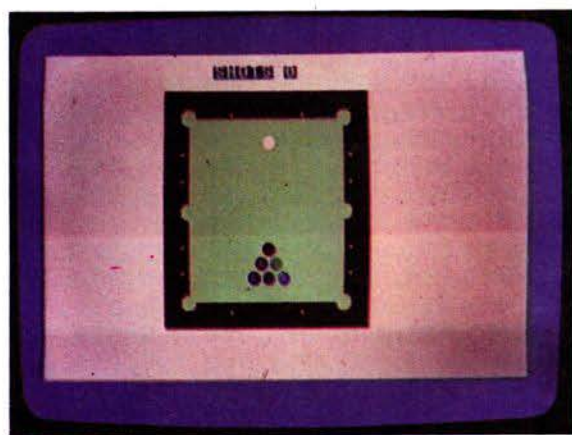


51702 :141,194,096,141,195,096,085
 51708 :162,016,078,086,064,110,000
 51714 :084,064,144,016,173,194,165
 51720 :096,024,109,085,064,141,015
 51726 :194,096,173,195,096,109,109
 51732 :087,064,074,141,195,096,165
 51738 :110,194,096,110,088,064,176
 51744 :110,089,064,202,208,216,153
 51750 :173,190,096,240,019,173,161
 51756 :088,064,032,018,201,141,076
 51762 :088,064,173,194,096,073,226
 51768 :255,105,000,141,194,096,079
 51774 :172,194,096,173,088,064,081
 51780 :096,032,195,203,169,000,251
 51786 :141,005,212,169,161,141,135
 51792 :006,212,169,001,141,024,121
 51798 :212,169,000,141,012,212,064
 51804 :169,161,141,013,212,160,180
 51810 :000,162,015,185,190,202,084
 51816 :141,000,212,010,141,014,110
 51822 :212,185,189,202,141,001,016
 51828 :212,042,141,015,212,185,155
 51834 :064,203,010,141,007,212,247
 51840 :185,063,203,042,141,008,002
 51846 :212,169,019,141,004,212,123
 51852 :169,033,141,011,212,165,103
 51858 :162,105,010,197,162,208,222
 51864 :252,169,017,141,004,212,179
 51870 :169,032,141,011,212,166,121
 51876 :203,189,129,235,201,049,146
 51882 :144,008,201,052,176,004,243
 51888 :141,163,096,096,200,200,048
 51894 :192,128,208,171,076,069,002
 51900 :202,026,156,026,156,022,008
 51906 :096,022,096,026,156,026,104
 51912 :156,022,096,026,156,026,170
 51918 :156,022,096,026,156,029,179
 51924 :223,033,135,029,223,026,113
 51930 :156,022,096,019,239,022,004
 51936 :096,022,096,016,195,016,153
 51942 :195,016,195,016,195,016,095
 51948 :195,016,195,019,239,022,154
 51954 :096,016,195,019,239,022,061
 51960 :096,022,096,016,195,019,180
 51966 :239,019,239,022,096,014,115
 51972 :239,014,239,014,239,014,251
 51978 :239,014,239,014,239,019,006
 51984 :239,022,096,016,195,019,091
 51990 :239,022,096,022,096,016,001
 51996 :195,019,239,019,239,022,249

52002 :096,013,078,013,078,013,069
 52008 :078,013,078,013,078,013,057
 52014 :078,000,000,026,156,000,050
 52020 :000,026,156,000,000,026,004
 52026 :156,000,000,026,156,004,144
 52032 :112,004,112,005,152,005,198
 52038 :152,005,152,005,152,003,027
 52044 :082,003,082,004,112,004,107
 52050 :112,005,152,005,152,003,255
 52056 :186,003,186,003,244,003,201
 52062 :244,004,048,004,048,004,190
 52068 :250,004,250,004,250,004,094
 52074 :250,002,124,002,124,003,099
 52080 :082,003,082,004,048,004,079
 52086 :048,003,082,003,082,003,083
 52092 :134,003,134,003,186,003,075
 52098 :186,004,112,004,112,004,040
 52104 :112,004,112,002,124,002,236
 52110 :124,003,186,003,186,004,136
 52116 :112,004,112,003,186,003,056
 52122 :186,003,244,003,244,004,070
 52128 :048,004,048,004,250,004,006
 52134 :250,004,250,004,250,002,158
 52140 :124,002,124,003,082,003,254
 52146 :082,003,083,003,083,003,179
 52152 :187,003,187,004,048,004,105
 52158 :048,004,112,004,112,160,118
 52164 :023,169,000,153,000,212,241
 52170 :136,016,250,096,169,147,248
 52176 :032,210,255,169,001,141,248
 52182 :033,208,032,076,204,169,168
 52188 :016,141,162,005,169,012,213
 52194 :141,165,005,169,160,141,239
 52200 :004,208,169,168,141,006,160
 52206 :208,162,045,160,215,142,146
 52212 :005,208,140,007,208,136,180
 52218 :232,165,162,197,162,240,128
 52224 :252,224,131,208,238,162,191
 52230 :013,160,013,024,032,240,232
 52236 :255,169,174,160,204,022,238
 52242 :030,171,162,015,160,013,057
 52248 :024,032,240,255,169,192,168
 52254 :160,204,032,030,171,162,021
 52260 :017,160,013,024,032,240,010
 52266 :255,169,209,160,204,032,047
 52272 :030,171,032,069,202,032,072
 52278 :195,203,173,163,096,201,061
 52284 :051,208,001,000,073,048,185
 52290 :240,244,201,003,176,240,146
 52296 :141,058,099,096,169,009,132
 52302 :160,000,153,000,216,153,248
 52308 :000,217,153,000,218,153,057
 52314 :000,219,136,208,241,096,222
 52320 :060,126,255,255,255,255,022
 52326 :126,060,150,150,085,085,246
 52332 :150,150,000,000,128,000,024
 52338 :038,000,000,038,255,255,188
 52344 :255,248,240,224,224,224,255
 52350 :255,255,255,031,015,007,176

52356 :007,007,224,224,224,240,034
 52362 :248,255,255,255,007,007,141
 52368 :007,015,031,255,255,255,194
 52374 :153,155,153,100,153,165,005
 52380 :148,174,158,174,143,183,112
 52386 :153,183,163,183,000,001,077
 52392 :002,004,005,006,012,014,211
 52398 :146,144,049,046,032,031,110
 52404 :032,079,078,069,032,080,038
 52410 :076,065,089,069,082,000,055
 52416 :144,050,046,032,031,032,015
 52422 :084,087,079,032,080,076,124
 52428 :065,089,069,082,000,144,141
 52434 :051,046,032,028,083,084,022
 52440 :079,080,032,080,076,065,116
 52446 :089,073,078,071,000,018,039
 52452 :031,080,076,065,089,069,126
 52458 :082,032,049,000,080,076,041
 52464 :065,089,069,082,032,146,211
 52470 :050,000,173,058,099,201,059
 52476 :001,208,031,162,001,142,029
 52482 :051,099,160,020,024,032,132
 52488 :240,255,238,059,099,208,083
 52494 :003,238,060,099,173,060,135
 52500 :099,174,059,099,032,205,176
 52506 :189,076,064,205,173,001,222
 52512 :088,208,005,173,164,096,254
 52518 :208,024,173,048,004,073,056
 52524 :128,141,048,004,173,076,102
 52530 :004,073,128,141,076,004,220
 52536 :173,051,099,073,001,141,082
 52542 :051,099,096,162,014,094,066
 52548 :000,096,094,016,096,202,060
 52554 :016,247,096,005,018,032,232
 52560 :032,032,032,032,032,032,016
 52566 :032,032,032,032,032,032,022
 52572 :032,032,032,032,032,032,028
 52578 :032,032,032,032,032,032,034
 52584 :032,032,032,032,032,032,040
 52590 :032,032,032,032,032,032,046
 52596 :032,032,032,000,005,018,235
 52602 :032,032,032,032,032,032,058
 52608 :032,032,032,032,029,029,058
 52614 :029,029,029,029,029,029,052
 52620 :029,029,029,029,029,029,058
 52626 :029,029,029,032,032,032,073
 52632 :032,032,032,032,032,032,088
 52638 :032,032,032,032,000,018,048
 52644 :031,083,072,079,084,083,084
 52650 :000,029,029,073,084,039,168
 52656 :083,032,065,032,084,073,033
 52662 :069,000,032,087,073,078,009
 52668 :083,000,152,170,094,000,175
 52674 :096,094,016,096,096,000,080
 52680 :028,057,085,114,142,170,028
 52686 :199,228,142,161,096,162,170
 52692 :000,189,002,206,157,000,254
 52698 :212,232,224,025,208,245,084
 52704 :169,016,141,004,212,169,167

52710 :128,141,011,212,169,016,139
 52716 :141,018,212,169,017,141,166
 52722 :004,212,169,129,141,011,140
 52728 :212,169,017,141,018,212,249
 52734 :174,161,096,096,100,007,120
 52740 :000,000,016,036,000,100,156
 52746 :003,000,000,129,016,000,158
 52752 :100,004,000,000,017,016,153
 52758 :000,000,000,000,015,162,199
 52764 :022,160,009,024,032,240,003
 52770 :255,173,060,099,208,045,106
 52776 :173,059,099,201,009,176,245
 52782 :010,169,152,160,206,032,007
 52788 :030,171,076,161,192,201,115
 52794 :012,176,010,169,132,160,205
 52800 :206,032,030,171,076,161,228
 52806 :192,201,021,176,010,169,071
 52812 :113,160,206,032,030,171,020
 52818 :076,161,192,169,095,160,167
 52824 :206,032,030,171,076,161,252
 52830 :192,029,029,089,079,085,085
 52836 :039,082,069,032,065,032,163
 52842 :078,079,086,073,067,069,046
 52848 :000,029,089,079,085,039,177
 52854 :082,069,032,065,078,032,220
 52860 :065,077,065,084,069,085,057
 52866 :082,000,089,079,085,039,248
 52872 :082,069,032,065,032,080,240
 52878 :079,079,076,032,083,072,051
 52884 :065,082,075,000,029,029,172
 52890 :029,089,079,085,039,082,045
 52896 :069,032,065,032,080,082,008
 52902 :079,000,013,016,255,255,016





Number Quest

di G. e B. Schueler
trad. ed adatt.
di M. Cristuib Grizzi

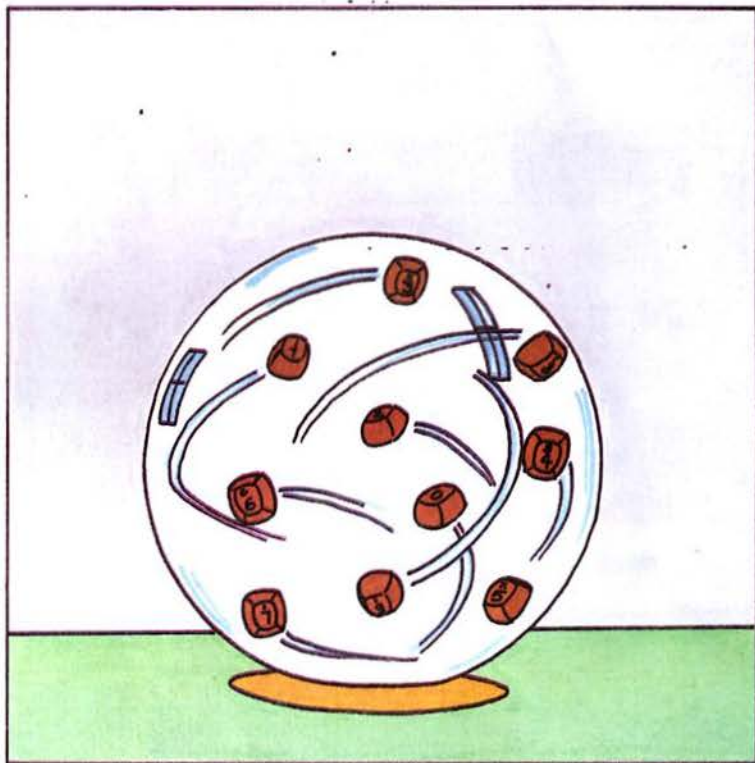
In questo gioco, studiato per esercitare i bambini nel calcolo aritmetico, solo rispondendo correttamente si riuscirà a far atterrare sano e salvo un paracadutista lanciatisi da un aereo.

Si dispone di cinque possibilità per indovinare un numero, scelto dal computer, compreso tra 3 e 100. Per ogni risposta sbagliata il computer fornisce un piccolo aiuto nella ricerca del numero.

Questo aiuto può consistere in un numero indicante quante volte il numero inserito è più grande o più piccolo di quello scelto dal computer. Ad esempio, se il numero corretto è 75, e si inserisce il numero 15, il computer risponderà "5 volte troppo basso". Tuttavia, se il numero che inserite differisce da quello scelto dal computer solo per un fattore 2 (o minore), verrà semplicemente visualizzato il messaggio "troppo alto" o "troppo basso", a seconda dei casi.

Se si indovina il numero generato dal computer entro i cinque tentativi, si potrà vedere il paracadutista discendere verso terra e compiere un atterraggio perfetto. Se invece non si riesce a raggiungere l'obiettivo, il paracadutista farà un bel tuffo nell'acqua del lago.

Se si desidera modificare il numero dei tentativi disponibili, è possibile, nella versione per C64, modificare il 5 in linea 250 e il valore di TN (=6) in linea 500. Questo 6 indica infatti che il valore di TN, la variabile che conteggia i tentativi, ha oltrepassato il valore limite di 5. Nella versione per VIC 20 si può modificare il 5 in linea 170 ed il valore di TN in linea 380.



Number Quest versione per VIC 20

```
10 PRINT "{CLR}{ 11 GIU' } { 5 DES } ATTENDERE
  .." :rem 9
20 POKE55,0:POKE56,28:CLR :rem 220
30 B=7168:CB=25600:FORJ=0TO511:POKEB+J,PE
  EK(B+J+CB):NEXTJ :rem 175
40 FORI=216TO255:READA%:POKEB+I,A%:NEXTI:
  FORI=280TO295:READA%:POKEB+I,A%:NEXTI:
  PRINT "{CLR}" :rem 41
```

```
50 POKE36869,240:SC=36879:POKESC,233 :rem 71
60 PRINTSPC(89)"[BLK]{ 2 SPAZI }[<M>]M
  [< 2 G>][< 2 M>]MN[<G>]O[<G>]O[<G>]" :rem 114
70 PRINT "{ 3 SPAZI }[<M>] [<G>]L@[<M>]
  { 2 SPAZI }[<G>]O[<G>]O[<G>]" :rem 13
80 PRINTTAB(12)"[<T>] [<T>]" :rem 10
90 PRINT "{RED}{ 5 SPAZI }OP [<G>][<M>] O
  O PO" :rem T26
```



```

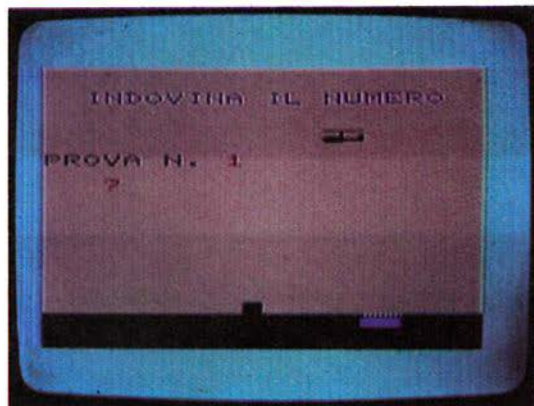
100 PRINT"{ 5 SPAZI}Le[<F>]Le O P [<M>]  
[<G>]" :rem 20  
110 PRINTTAB(11)"[<T>] [<T>]" :rem 51  
120 V=36878:F2=36875:POKEV,15 :rem 131  
130 READA%,D%:IFA%=-1THEN150 :rem 58  
140 POKEF2,A%:FORT=1TOD%:NEXTT:GOTO130 :rem 145  
150 POKEV,0:POKEF2,0 :rem 180  
160 FORT=1TO1000:NEXTT :rem 113  
170 PRINT"[CLR]":POKESC,26:PRINT"{BLK}HAI  
{RED}5 {BLK}POSSIBILITA'" :rem 71  
180 PRINT:PRINT"PER INDOVINARE UN  
{ 5 SPAZI}NUMERO" :rem 128  
190 PRINT:PRINT"{BLK}DA {RED}10 {BLK} A  
{RED}100":FORT=1TO4000:NEXTT:rem 89  
200 PRINT"[CLR]":POKESC,27:TN=1:POKE36869  
,255 :rem 21  
210 PRINTSPC(46)"{BLU}INDOVINA IL NUMERO"  
 :rem 231  
220 PRINT:PRINT:PRINTTAB(14)"{BLK}[E]:PRI  
NT"{ 12 GIU'}":PRINTTAB(10)"{BLK}]" :rem 31  
230 PRINT"{BLK}}]]]]]]]]]]]]]]]]]]]]]{BLU}↑↑  
{BLK}}]]]]";P=PRINT"]]" :rem 224  
240 PRINT"}]]]]]]]]]]]]]]]]]]]]]";c=POKE8163,2  
9:POKE38883,0 :rem 243  
250 FORN=0TO21:POKE8164+N,29:POKE38884+N,  
0:NEXTN :rem 159  
260 PRINT"{ 16 SU}" :rem 121  
270 R=INT(RND(1)*90+10) :rem 233  
280 PRINT"[BLK]PROVA N."; "{RED}";TN:PRINT  
 :rem 62  
290 S$="":INPUT"{RED}{ 3 SPAZI}";S$ :rem 111  
300 S=VAL(S$):IFS<LORS>100THENGOSUB570 :rem 101  
 :rem 113  
310 FORU=1TOLEN(S$):UQ=ASC(MID$(S$,U,1)) :rem 113  
320 IFUQ<48ORUQ>57THENU=LEN(S$):NEXT:GOSU  
B570:GOTO280 :rem 167  
330 NEXT :rem 213  
340 IFS>2THEN370 :rem 178  
350 PRINT" DA 3 A 100!{ 3 SPAZI}";:FORT=0  
TO1000:NEXT :rem 14  
360 PRINT"{ 15 SIN}{ 15 SPAZI}{ 2 SU}":GO  
TO290 :rem 205  
370 IFS=RTHEN650 :rem 213  
380 TN=TN+1:IFTN=6THEN410 :rem 26  
390 IFS>RTHEN490 :rem 218  
400 IFS<RTHEN510 :rem 201  
410 POKEV,15:FORM=220TO190STEP-1:POKEF2,M  
:FORT=1TO20:NEXTT:NEXTM:POKEF2,0:POKE  
V,0 :rem 239  
420 P=14:GOSUB580:POKEX,36:POKECX,6:POKE  
+22,30:POKECX+22,6 :rem 111  
430 POKE36877,220:FORL=15TO5STEP-1:POKEV,  
L:FORM=1TO50 :rem 50  
440 NEXTM:NEXTL:POKE36877,0:POKEV,0  
 :rem 216  
450 PRINT"[CLR]":POKESC,216:PRINTSPC(157)  
"{BLK}HAI PERSONI!!{ 4 SPAZI}" :rem 238  
460 PRINTSPC(46)"IL NUMERO ERA: {RED}";R  
 :rem 50  
470 PRINT:FORT=1TO1000:NEXTT :rem 60  
480 GOTO700 :rem 108  
490 D=INT(S/R):D$="ALTO":IFD<=1THEN560  
 :rem 2  
500 GOTO520 :rem 101  
510 D=INT(R/S):D$="BASSO":IFD<=1THEN560  
 :rem 67  
520 PRINT"{RED} ";D;"{BLK} VOLTE  
{ 5 SPAZI}";"{RED}";D$ :rem 218

```

```

530 POKEV,15:FORM=250TO180STEP-1:POKEF2,M
: NEXTM: FORT=1TO1000: NEXT: POKEV,0
: rem 194
540 PRINT"{ 2 SU}{ 10 SPAZI}" : rem 140
550 PRINT"{ 21 SPAZI}":PRINT" { 5 SU}":GOT
O280 : rem 88
560 PRINT"{RED}{ 5 SPAZI}";"{BLU}";D$:FOR
T=1TO400: NEXT: GOTO530 : rem 103
570 PRINT(RED) DA 1 A 100 ":FORT=1TO800
NEXTT:PRINT"{SU}{ 11 SPAZI}":PRINT"
{ 5 SU}":GOTO280 : rem 133
580 X=7680+(5*22)+17: CX=X+30720: Z=7680+(5
*22)+15: CZ=Z+30720 : rem 117
590 FORN=1TOP: POKEZ,32: POKEZ+1,32: POKEZ,3
2: POKEZ+22,32 : rem 107
600 POKEZ-1,27: POKEZ-1,0: POKEZ,28: POKEZ
,0 : rem 141
610 POKEZ+22,31: POKECX+22,0: POKEZ+44,35:P
OKECX+44,0 : rem 7
620 Z=Z-1: CZ=CZ-1: X=X+22: CX= CX+22: NEXTN: R
ETURN : rem 201
630 BL=36865: FORN=1TO6: POKEBL,135: FORT=1T
O50: NEXTT : rem 154
640 POKEBL,25: FORT=1TO40: NEXTT: NEXTN: RETU
RN : rem 130
650 GOSUB630: P=5: GOSUB580: FORN=1TO7
: rem 190
660 POKEZ,32: POKEZ+22,32 : rem 149
670 POKEZ+21,31: POKECX+21,0: POKEZ+43,35: P
OKECX+43,0: X=X+21: CX= CX+21: NEXTN
: rem 192
680 POKEV,15: FORL=130TO240: POKEF2,L: NEXTL
: POKEV,0: FORT=1TO1000: NEXTT : rem 121
690 PRINT"{CLR}":POKESC,26: PRINTSPC(157)"
CONGRATULAZIONI": PRINTSPC(69)"HAI IND
OVINATO!" : rem 141
700 FORT=1TO200: NEXTT: GOTO200 : rem 119
710 DATA 255,1,1,1,127,255,255,127,255,0,
0,3,3,255,255,0 : rem 171
720 DATA 255,255,255,255,255,255,255,255,
170,170,255,255,255,255,255,255
: rem 255
730 DATA 60,126,255,255,129,66,36,24,16,1
8,60,88,40,36 : rem 113
735 DATA 66,0,17,130,68,37,6,148,93,126
: rem 141
740 DATA 231,200,235,450,231,200,232,200,
235,200,232,200,231,200,235,500,-1,-1
: rem 181

```




```

10 PRINT"{CLR}{ 10 DES}{ 12 GIU'";;"ATTEN  
DERE PREGO..." :rem 213
20 POKE56,48:CLR:K=12288:PC=56334:CP=5327  
2 :rem 146
30 IF PEEK(12288)+PEEK(12289)=162THEN60 :rem 127
40 POKEPC,PEEK(PC)AND254:POKE1,PEEK(1)AND  
251 :rem 161
50 FORJ=0TO4095:POKEK+J,PEEK(53248+J):NEX  
TJ :rem 129
60 FORI=216TO255:READA$:POKEK+I,A$;NEXTI:  
FORI=280TO295:READA$:POKEK+I,A$;NEXTI :rem 159
70 POKE1,PEEK(1)OR4:POKEPC,PEEK(PC)OR1 :rem 114
80 POKECP,21:BC=53280:BK=53281:POKEBC,2:P  
OKEBK,1:PRINT"{CLR}" :rem 251
90 PRINTSPC(90)"{BLK}{ 2 SPAZI}[<M>]M  
[< 2 G>][< 2 M>]MN[<G>]O[<G>]OO[<G>]" :rem 109
100 PRINT"{ 12 SPAZI}[<M>][<G>]L[<M>  
{ 2 SPAZI}[<G>]O[<G>]OO[<G>]" :rem 55
110 PRINTTAB(21)"[<Y>][<Y>]" :rem 92
120 PRINT:PRINT"{RED}{ 13 SPAZI}OP[<G>]  
[<M>] O O PO" :rem 111
130 PRINT"{ 13 SPAZI]L[<F>]L O P [<M>]  
[<G>]" :rem 23
140 PRINTTAB(19)"[<Y>][<Y>]" :rem 102
150 PRINTSPC(136)"{BLU}PREMI F1" :rem 75
160 V=54272:FORSL=VTOV+24:POKESL,0:NEXT :rem 232
170 POKEV+5,88:POKEV+4,33:POKEV+24,143:PO  
KEV+6,195 :rem 200
180 READLF,HF,DR:IFLF=-1THEN200 :rem 122
190 POKEV,LF:POKEV+1,HF:FORT=1TODR:NEXTT:  
GOTO180 :rem 167
200 POKEV,0:POKEV+1,0:POKEV+24,0:rem 150
210 GETAS:IFAS=""THEN210 :rem 73
220 IFAS=CHR$(133)THEN240 :rem 115
230 GOTO210 :rem 97
240 PRINT"{CLR}":POKEBC,7:POKEBK,15 :rem 211
250 PRINTSPC(211)"{BLK}DISPONI DI {RED}5  
{BLK} TENTATIVI":PRINTSPC(51)"PER IND  
OVINARE UN" :rem 32
260 PRINTSPC(51)"{BLK}NUMERO TRA  
{ 2 SPAZI}{RED}3{BLK} E {RED}100":PRI  
NTSPC(211)"{BLU}PREMI F1" :rem 13
270 GETAS:IFAS=""THEN270 :rem 85
280 IFAS=CHR$(133)THEN300 :rem 118
290 GOTO270 :rem 109
300 PRINT"{CLR}":POKEBC,6:POKEBK,1:TW=1:P  
OKECP,(PEEK(CP)AND240)+12 :rem 93
310 PRINTSPC(92)"{BLU}INDOVINA IL NUMERO"  
 :rem 233
320 PRINT:PRINTTAB(24)"{BLK}{£":PRINT"  
{ 12 GIU'":PRINTTAB(19)"{BLK}" :rem 99
330 PRINT"{BLK}}]]]]]]]]]]]]]]]]]]]]]]]]  
{BLU}↑↑↑↑{BLK}}]]]]]]]]]]]]]]]]]]]]]]  
]]]]]]]]"; :rem 177
340 FORT=1TO2 :rem 21
350 PRINT"{BLK}}]]]]]]]]]]]]]]]]]]]]]]]]  
]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]  
; :rem 186
360 NEXT :rem 216
370 FORN=0TO40:POKE1983+N,29:POKE56255+N,  
0:NEXTN :rem 157
380 PRINT"{ 15 SU}" :rem 235

```

```

390 R=INT(RND(1)*90+10) :rem 236
400 PRINT"[BLK]TENTATIVO: ";"[RED]";TN:PR :rem 44
INT :rem 44
410 PRINT"{ 5 SPAZI}"; :rem 161
420 SS="":INPUTSS:S=VAL(SS):IFS<1ORS>100T :rem 60
HENGOSUB690:PRINT"{GIU'}";:GOTO400 :rem 119
430 FORU=1TOLEN(SS):UT=ASC(MID$(SS,U,1)) :rem 4
440 IFUT<48ORUT>57THENU=LEN(SS):GOSUB690: :rem 216
NEXT:PRINT"{GIU'}";:GOTO400 :rem 18
450 NEXT :rem 18
460 IFS>2THEN490 :rem 31
470 PRINT"[BLK]{ 2 SPAZI}DA 3 A 100 :rem 106
{ 5 SPAZI":FOR T=0TO1000:NEXT:PRINT" :rem 220
{SU}{ 18 SPAZI}{ 2 SU}{RED}": :rem 23
480 GOTO410 :rem 206
490 IFS=RTHEN780 :rem 207
500 TN=TN+1:IFTN=6THEN530 :rem 66
510 IFS>RTHEN610 :rem 209
520 IFS<RTHEN630 :rem 110
530 POKEV+24,143:FORM=60TO30STEP-2:POKEV, :rem 110
INT(M/3):POKEV+1,M:FORT=1TO10:NEXTT :rem 110
:rem 255
540 NEXTM:POKEV+24,0 :rem 107
550 P=14:GOSUB710:POKEK,36:POKECX,6:POKEK :rem 73
+40,30:POKECX+40,6 :rem 193
560 FORM=125TO15STEP-5:POKEV+24,M:FORT=1T :rem 33
O20:NEXTT:NEXTM:POKEV+24,0 :rem 163
570 PRINT"{CLR}":POKESC,216:PRINTSPC(210) :rem 230
"{BLK}SPIACENTE, HAI PERSO" :rem 81
580 PRINTSPC(130)"IL NUMERO ERA{RED} ";R :rem 193
:rem 78
590 PRINT"[BLK]{GIU'}AVRAI MIGLIOR FORTUN :rem 108
A LA PROSSIMA VOLTA":FORU=1TO1200:NEX :rem 255
TU :rem 107
600 GOTO850 :rem 73
610 D=INT(S/R):D$="ALTO":IFD<=1THEN680 :rem 193
:rem 33
620 GOTO640 :rem 107
630 D=INT(R/S):D$="BASSO":IFD<=1THEN680 :rem 73
:rem 193
640 PRINT"{RED} ";D;"[BLK]VOLTE TROPPO "; :rem 33
"[RED]";D$ :rem 107
650 POKEV+24,143:FORM=50TO0STEP-1:POKEV,I :rem 230
NT(M/2):POKEV+1,M:NEXTM :rem 81
660 FORT=1TO600:NEXT:POKEV+24,0 :rem 193
670 PRINT"{SU}{ 23 SPAZI}":PRINT"{ 5 SU} :rem 230
":GOTO400 :rem 193
680 PRINT"{RED} TROPPO ";"[BLU]";D$:FORT= :rem 81
1TO400:NEXT:GOTO650 :rem 193
690 PRINT"{RED}{ 3 SPAZI}PROVA 3-100 :rem 240
{ 2 SPAZI}":FORT=1TO800:NEXTT :rem 46
:rem 100
700 PRINT"{SU}{ 14 SPAZI}":PRINT"{ 5 SU} :rem 102
";:RETURN :rem 100
710 X=1024+(5*40)+26:CX=X+54272:Z=1024+(5 :rem 102
*40)+24:CZ=Z+54272 :rem 145
720 FORN=1TOP:POKEZ,32:POKEZ+1,32:POKEK,3 :rem 13
2:POKEK+40,32 :rem 205
730 POKEZ-1,27:POKEK-1,0:POKEZ,28:POKEKZ :rem 115
,0 :rem 43
740 POKEK+40,31:POKECX+40,2:POKEK+80,35:P :rem 199
OKECX+80,0 :rem 199
750 Z=Z-1:CZ=CZ-1:X=X+40:CX=CX+40:NEXTN:R :rem 115
ETURN :rem 43
760 BL=53265:FORN=1TO6:POKEBL,PEEK(BL)AND :rem 199
239:FORT=1TO50:NEXTT :rem 115
770 POKEBL,PEEK(BL)OR16:FORT=1TO40:NEXTT :rem 43
NEXTN:RETURN :rem 199
780 GOSUB760:P=5:GOSUB710:FORN=1TO7 :rem 199

```




Heat seeker

di J. Wolverton
trad. ed adatt.
di M. Cristuib Grizzi

Il vostro jet compie una rapida cabrata per evitare il missile, quindi esegue una veloce picchiata. Nulla da fare, lo avete sempre in coda, non potete evitare un missile guidato dal calore dei vostri motori! Ecco un veloce gioco d'azione per il VIC 20 inespanso ed il C64.

I missili termoguidati sono molto pericolosi: rilevano il calore generato dai motori degli aerei e puntano su di essi, seguendoli in tutte le loro manovre di sganciamento fino a che non raggiungono il bersaglio, essendo più veloci dei jet stessi.

Il vostro compito: distruggere la base dei missili termoguidati. È abbastanza facile colpire i missili a terra, ma, una volta lanciati, dovrete prodigarvi in manovre evasive per liberarvene. Come se non bastasse, il cielo è pieno di mine aeree...

Pilotare il jet

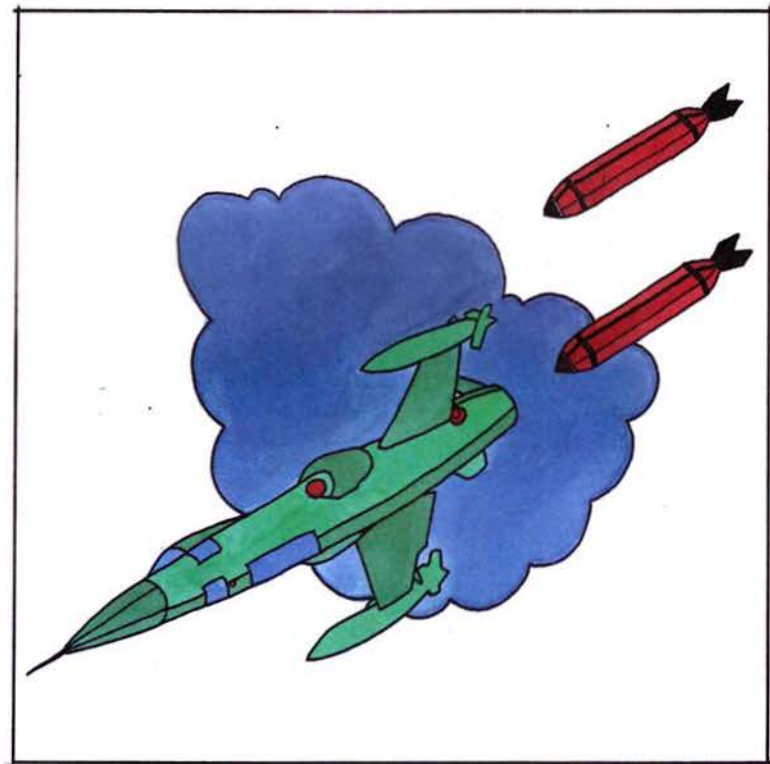
Il vostro jet è controllato dal joystick: come una cloche di aereo, se lo tirate verso voi, l'aereo riprende quota, mentre se lo spostate in avanti, l'aereo esegue una picchiata. Il jet si muove a velocità costante — non è possibile accelerare o rallentare — e con il pulsante di fuoco potete lanciare un missile contro i vostri obbiettivi sul terreno.

Nella versione per VIC 20 potete inoltre sparare contro il missile termoguidato che vi segue e potete sparare contro le mine aeree che vi bloccano la strada. È anche possibile sparare accidentalmente contro se stessi.

Disponete di otto jet, e potete fermare il gioco in qualsiasi momento premendo SHIFT LOCK.

Note alla versione per C64

Il jet ed i missili termoguidati sono degli SPRITE, dotati quindi di movimento pixel per pixel. Potete sparare contro i missili a terra, ma non potete sparare contro i missili che vi inseguono; essi



sono infatti dotati di un circuito IEE (Improved Electronic Evasion) che li rende praticamente impossibili da colpire.

L'unico modo per neutralizzare un missile inseguitore è manovrare in modo da farlo schiantare sul terreno, eseguendo una picchiata e riprendendo l'aereo all'ultimo secondo prima che si sfasci al suolo. I missili termoguidati sono infatti molto veloci, ma non hanno capacità di manovra rapida.

Al contrario della versione per VIC 20, non vi può capitare di sparare contro voi stessi e non ci sono mine ad ostacolare le manovre. Se il vostro jet viene distrutto, riappaiono tutti i missili a ter-

ra (al contrario della versione per VIC 20, che mantiene il numero dei missili distrutti).

Si può giocare anche in due, in modo non competitivo: i due giocatori pilotano il jet a turno, cercando di ottenere un punteggio migliore di quello dell'avversario. Il programma legge entrambe le porte joystick, quindi il giocatore correntemente "inattivo" non dovrà toccare il proprio joystick, al fine di evitare interferenze.

Esistono tre livelli di difficoltà: novizio, intermedio ed esperto, ed un bonus di dieci punti viene regalato ogni pochi secondi mentre si è in volo, solo perché

si riesce a rimanere in aria.

Istruzioni speciali

Per il C64, se si dispone della cassetta allegata alla rivista, è sufficiente posizionarsi con il nastro in corrispondenza dell'inizio del programma e digitare:

LOAD

Il programma andrà quindi mandato in esecuzione digitando:

RUN

Ricordiamo che per conoscere il punto esatto di inizio dei vari programmi su cassetta è sufficiente inserire quest'ultima completamente riavvolta nel registratore, azzerare il contanastro e digitare:

LOAD***

Dal momento che sicuramente non esiste sulla cassetta alcun programma di nome *** il computer passerà in rassegna tutti i programmi incontrati, visualizzandone il nome con il messaggio:

FOUND XXXX

Sarà quindi utile annotarsi su un foglio i nomi dei programmi e la relativa posizione del contanastro (sottrarre tre o quattro unità dal valore corrispondente all'apparizione del messaggio FOUND...).

Gli abbonati alla sola rivista dovranno invece usare il programma MLX (vedere nelle pagine verdi) per digitare il lista- to in linguaggio macchina. La procedura da seguire, in questo caso, è:

1. Spegner e riaccendere il computer e digitare in modo diretto le seguenti istruzioni:

POKE642,32:SYS58260

2. Caricare in memoria MLX (che sarà stato precedentemente digitato e salvato), e rispondere alle sue richieste circa gli indirizzi iniziale e finale con:

indirizzo iniziale: 2049

indirizzo finale: 6470

Dopo avere digitato e salvato il programma tramite MLX si potranno seguire per il caricamento le istruzioni date più sopra per i possessori di cassetta. La versione per VIC 20 va invece caricata od eventualmente digitata come un qualsiasi programma BASIC.

Heat seeker per VIC 20

```

100 GOTO500                                :rem 95
110 GOSUB650:IFPEEK(653)=1THEN110          :rem 185
120 FORR=0TO7:J=PEEK(37137)AND44:IFJ<>44T :rem 73
    HEN380                                  :rem 72
130 IFC=7679THEN180                         :rem 52
140 POKEC,32:C=C+1                          :rem 4
150 IFC>8163ORC<7724THENC=7679:GOTO180    :rem 21
160 GE=PEEK(C):IFGE<>32THEN450              :rem 125
170 POKEC,G                                :rem 194
180 IFR=20RR=6THENNEXT:GOTO110             :rem 156
190 POKEB,32                               :rem 209
200 B=B+D(B1):IFB=ATHEN300                 :rem 234
210 IFPEEK(B)=16ORB<7724THENB1=B1+4+(B1>4 :rem 14
    )#8:GOTO200
220 IFPEEK(B)=0THEN430                     :rem 4
230 POKEB,17:IFR=INT(R/2)*2THEN260         :rem 123
240 Y=INT(D(B1)/21+.5):X=D(B1)-22*Y        :rem 89
250 V=INT((A-B)/21+.5):H=A-B-V*22:B1=B1+S :rem 230
    GN(X*V-Y*H):B1=B1+((B1>8)-(B1<1))*8
260 IFR=0THENNEXT:GOTO110                  :rem 247
270 POKEA,32:A=A+D(A1):IFPEEK(A)<>32THEN3    :rem 137
    00                                       :rem 170
280 IFA<7724THENA1=A1+4+8*(A1>4):GOTO270   :rem 104
290 POKEA,A1+7:NEXT:GOTO110                :rem 151
300 AE=PEEK(A):POKEA,18:GOSUB620:POKEB,32 :rem 242
    :POKEC,32:IFA<8164THENPOKEA,32:GOTO32 :rem 17
    0                                       :rem 163
310 POKEA,16                               :rem 170
320 IFAE=0THENK=K+1:IFK=11THEN560          :rem 107
330 L=L-1:IFL>0THEN600                     :rem 171
340 GOSUB650:POKE198,0:PRINT"HOME"        :rem 171
    {10GIU'}{3DES}{RVS}GIOCHI ANCORA?    :rem 104
    (S/N)"                                  :rem 51
350 GETAS:IFAS="S"THEN500                  :rem 28
360 IFAS="N"THENSYS58648:END               :rem 38
370 GOTO350                                :rem 27
380 IF(JAND8)=0THENA1=A1-1-8*(A1=1)        :rem 38
390 IF(JAND4)=0THENA1=A1+1+8*(A1=8)        :rem 27
400 IFJAND32THEN130                        :rem 192

```

```

410 POKEC,32:C1=D(A1):C=A+C1+C1:G=A1:IFG= :rem 95
    8THENG=4                                :rem 57
420 POKEVO,15:POKES2,190:POKES4,180:FOR= :rem 118
    1TO20:NEXT:POKEVO,0:GOTO150            :rem 248
430 SC=SC+50:K=K+1:IFK=11THEN560:rem      :rem 251
440 POKEB,18:GOSUB620:GOTO230              :rem 149
450 POKEC,18:GOSUB620:IFGE=0THENS=SC+50:  :rem 129
    K=K+1:IFK=11THENGOTO560                :rem 34
460 IFGE=17THENS=SC+100:POKEB,32:B=7905:  :rem 67
    B1=3                                     :rem 194
470 IFGE=19THENS=SC+10                     :rem 87
480 IFA=CTHENS=SC-100:A=7910:A1=3:POKEA,  :rem 129
    32                                       :rem 186
490 POKEC,32:C=7679:GOTO180                :rem 24
500 POKE52,28:POKE56,28:POKE36879,236:POK :rem 138
    E36869,240:CLR                          :rem 245
510 GOSUB630:PRINT"{10GIU'}{5SPAZI}"      :rem 171
    {RVS}HEAT SEEKER"                      :rem 42
520 FORA=7168TO7327:READJ:POKEA,J:NEXT    :rem 133
530 FORI=7336TOI+7:POKEI,PEEK(I-16):POKEI :rem 170
    +88,0:NEXT:POKE36869,255               :rem 243
540 VO=36878:S2=36876:S4=36877:FOR=1TO8:  :rem 80
    READD(T):NEXT                          :rem 157
550 L=8:SC=0:K=0                           :rem 51
560 IFK=11THENL=L+1:K=0:SC=SC+1000        :rem 28
570 GOSUB630:FOR=8142TO8163STEP2:POKET,0 :rem 38
    :NEXT                                    :rem 27
580 FOR=8164TO8185:POKET,16:NEXT          :rem 38
590 FOR=1TORND(1)*4:POKE7724+INT(RND(1)* :rem 133
    400),19:NEXT                           :rem 170
600 IFA<8164THENPOKEA,19                   :rem 243
610 A=7910:A1=3:C=7679:B=7905:B1=3:POKEA, :rem 80
    A1+7:GOTO110                             :rem 157
620 POKES4,190:FOR=100TO1STEP-5:POKEVO,T :rem 51
    /7:NEXT:RETURN                          :rem 28
630 PRINT"CLR">{BLK}";FORI=0TO483:PRINT"  :rem 38
    ";NEXT:PRINT"RED}";FORI=0TO20:PRIN :rem 27
    T";NEXT                                  :rem 38
640 POKE38905,2:PRINT"HOME">{BLK}";:RETU :rem 51
    N                                         :rem 28
650 PRINT"HOME">{RVS}PUNTI";SC;"HOME}";  :rem 38
    SPC(13);"NAVI";L:RETURN                 :rem 27
660 DATA 24,36,24,24,24,60,90,66:rem      :rem 38
670 DATA 24,24,24,24,24,24,24,24:rem      :rem 27
680 DATA 3,7,14,28,56,112,224,192        :rem 38

```



```

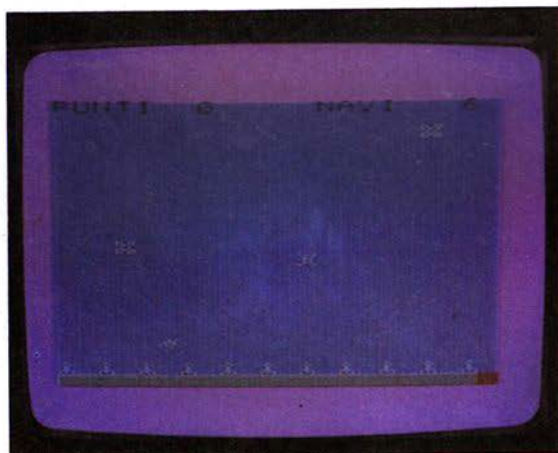
690 DATA 0,0,0,255,255,0,0,0 :rem 69
700 DATA 192,224,112,56,28,14,7,3 :rem 81
710 DATA 24,24,24,24,24,24,24,24 :rem 22
720 DATA 3,7,14,28,56,112,224,192 :rem 83
730 DATA 0,0,0,255,255,0,0,0 :rem 64
740 DATA 68,68,84,40,16,40,84,16 :rem 51
750 DATA 8,16,32,57,234,92,112,144 :rem 139
760 DATA 0,0,71,40,212,40,71,0 :rem 168
770 DATA 144,112,92,234,57,32,16,8 :rem 141
780 DATA 16,84,40,16,40,84,68,68 :rem 55
790 DATA 9,14,58,87,156,4,8,16 :rem 215
800 DATA 0,0,226,20,43,20,226,0 :rem 213
810 DATA 16,8,4,156,87,58,14,9 :rem 208
820 DATA 255,255,255,255,255,255,255,255 :rem 200
830 DATA 129,90,36,60,36,24,36,195 :rem 148
840 DATA 99,140,34,74,66,36,145,194 :rem 208
850 DATA 231,165,219,36,36,219,165,231 :rem 89
860 DATA -22,-21,1,23,22,21,-1,-23 :rem 90

```

```

2121 :000,141,015,212,141,014,084
2127 :212,169,032,141,019,212,096
2133 :169,127,141,020,212,169,155
2139 :129,141,018,212,169,001,249
2145 :141,003,212,169,025,141,020
2151 :005,212,169,000,141,025,143
2157 :008,032,244,020,032,108,041
2163 :019,169,048,160,006,153,158
2169 :200,007,136,208,250,140,038
2175 :021,008,172,248,020,048,132
2181 :018,160,006,153,225,007,190
2187 :136,208,250,169,050,141,069
2193 :198,007,169,049,141,223,164
2199 :007,169,252,141,017,008,233
2205 :169,011,162,004,157,050,198
2211 :017,232,232,224,016,208,068
2217 :247,032,141,013,169,008,011
2223 :141,022,008,141,023,008,006
2229 :076,075,011,169,000,141,141
2235 :066,017,141,067,017,032,015
2241 :111,013,173,084,017,201,024
2247 :255,208,034,032,074,013,047
2253 :173,212,014,201,008,144,189
2259 :004,201,248,144,020,173,233
2265 :213,014,201,008,144,004,033
2271 :201,248,144,009,032,084,173
2277 :013,032,135,013,076,034,020
2283 :011,173,066,017,240,003,233
2289 :032,145,010,120,169,253,202
2295 :141,000,220,173,001,220,234
2301 :041,128,240,243,169,247,041
2307 :141,000,220,088,169,004,113
2313 :141,018,008,162,000,189,015
2319 :068,017,201,127,144,006,066
2325 :173,018,008,032,122,010,128
2331 :014,018,008,232,224,006,017
2337 :208,235,165,161,205,020,003
2343 :008,240,006,141,020,008,206
2349 :032,127,012,173,084,017,234
2355 :016,033,201,192,240,029,250
2361 :201,255,240,025,032,002,044
2367 :012,144,007,169,192,141,216
2373 :084,017,208,013,169,255,047
2379 :141,084,017,169,128,141,243
2385 :212,014,141,213,014,160,067
2391 :009,169,255,217,074,017,060
2397 :240,013,136,208,248,173,087
2403 :084,017,201,192,208,003,036
2409 :076,018,011,173,031,208,110
2415 :141,016,008,041,001,240,046
2421 :009,032,071,010,032,084,099
2427 :013,076,075,011,173,016,231
2433 :008,041,002,240,003,032,199
2439 :033,010,173,016,008,041,160
2445 :252,208,003,076,192,008,112
2451 :141,016,008,169,004,141,114
2457 :019,008,170,173,019,008,038

```



Heat Seeker versione per C64

```

2049 :011,008,001,000,158,050,229
2055 :048,054,049,000,000,000,158
2061 :076,027,008,000,000,000,124
2067 :000,000,000,000,000,000,019
2073 :000,000,169,014,141,033,126
2079 :208,169,002,141,032,208,023
2085 :160,024,169,000,153,255,030
2091 :211,136,208,250,169,002,251
2097 :141,023,212,169,031,141,254
2103 :024,212,169,008,141,022,119
2109 :212,169,003,141,008,212,038
2115 :169,061,141,012,212,169,063

```


2463 :045,016,008,240,003,032,247
 2469 :179,009,014,019,008,232,114
 2475 :232,224,016,208,236,076,139
 2481 :192,008,045,016,208,240,118
 2487 :002,056,036,024,189,000,234
 2493 :208,106,056,233,008,176,208
 2499 :002,169,000,201,160,144,103
 2505 :002,169,144,074,074,074,226
 2511 :074,168,185,074,017,201,158
 2517 :255,208,072,169,192,153,238
 2523 :074,017,169,000,157,034,158
 2529 :017,157,035,017,169,226,078
 2535 :157,001,208,189,000,208,226
 2541 :056,233,016,041,224,024,063
 2547 :105,028,157,000,208,032,005
 2553 :154,012,138,074,170,169,198
 2559 :064,157,066,017,169,255,215
 2565 :157,248,007,138,010,170,223
 2571 :032,063,013,152,010,010,035
 2577 :168,169,096,153,113,007,211
 2583 :153,114,007,153,153,007,098
 2589 :153,154,007,096,173,084,184
 2595 :017,201,255,208,030,032,010
 2601 :181,012,032,063,013,169,255
 2607 :000,141,036,017,141,037,163
 2613 :017,141,084,017,169,226,195
 2619 :141,003,208,169,002,141,211
 2625 :040,208,032,084,013,096,026
 2631 :169,000,141,034,017,141,061
 2637 :035,017,169,226,141,001,154
 2643 :208,169,002,141,039,208,082
 2649 :169,001,013,028,208,141,137
 2655 :028,208,032,135,013,032,031
 2661 :063,013,160,192,132,162,055
 2667 :173,031,208,041,002,240,034
 2673 :003,032,033,010,164,162,005
 2679 :208,242,096,013,017,008,191
 2685 :141,017,008,173,018,008,234
 2691 :073,255,045,021,208,141,106
 2697 :021,208,169,254,157,250,172
 2703 :007,096,173,067,017,201,192
 2709 :028,176,006,169,000,141,157
 2715 :066,017,096,173,017,008,020
 2721 :208,001,096,169,000,141,008
 2727 :066,017,141,067,017,169,132
 2733 :004,170,168,045,017,008,073
 2739 :208,010,152,010,168,232,191
 2745 :232,224,016,208,242,096,179
 2751 :141,018,008,013,021,208,088
 2757 :141,021,208,173,018,008,254
 2763 :073,255,168,045,017,008,001
 2769 :141,017,008,173,016,208,004
 2775 :041,001,240,012,173,018,188
 2781 :008,013,016,208,141,016,111
 2787 :208,076,238,010,152,045,188
 2793 :016,208,141,016,208,173,227
 2799 :034,017,157,034,017,173,159



2805 :035,017,157,035,017,173,167
 2811 :000,208,157,000,208,173,229
 2817 :001,208,157,001,208,138,202
 2823 :074,168,169,000,153,066,125
 2829 :017,088,076,090,013,174,215
 2835 :021,008,254,022,008,160,236
 2841 :010,032,181,012,136,208,092
 2847 :250,240,041,169,000,141,104
 2853 :034,017,141,036,017,141,167
 2859 :035,017,141,037,017,032,066
 2865 :063,013,173,028,208,009,031
 2871 :001,141,028,208,169,002,092
 2877 :141,039,208,141,040,208,070
 2883 :169,192,133,162,165,162,026
 2889 :208,252,120,169,100,141,039
 2895 :000,208,169,100,141,001,186
 2901 :208,169,000,141,016,208,059
 2907 :169,001,141,021,208,169,032
 2913 :240,141,248,007,169,015,149
 2919 :141,039,208,169,254,045,191
 2925 :028,208,141,028,208,169,123
 2931 :000,133,160,133,161,133,067
 2937 :162,141,020,008,032,111,083
 2943 :019,162,009,169,255,157,130
 2949 :074,017,202,016,250,169,093
 2955 :000,141,084,017,173,031,073
 2961 :208,173,030,208,044,248,032
 2967 :020,048,028,160,009,185,089
 2973 :197,007,170,185,222,007,177
 2979 :153,197,007,138,153,222,009
 2985 :007,136,208,239,169,001,161
 2991 :056,237,021,008,141,021,147
 2997 :008,174,021,008,189,022,091
 3003 :008,208,019,160,000,044,114
 3009 :248,020,048,001,200,185,127
 3015 :022,008,208,202,136,016,023
 3021 :248,076,188,012,222,022,205
 3027 :008,189,022,008,024,105,055
 3033 :049,141,214,007,173,001,034
 3039 :220,045,000,220,041,016,253
 3045 :208,246,173,001,220,045,098
 3051 :000,220,041,016,240,246,230
 3057 :169,000,141,035,017,141,232
 3063 :036,017,169,085,141,034,217
 3069 :017,088,076,184,008,160,018
 3075 :009,185,074,017,201,255,232
 3081 :240,013,136,016,246,169,061
 3087 :253,045,021,208,141,021,192

3093 :208,056,096,169,192,153,127
 3099 :074,017,152,010,010,168,202
 3105 :169,096,153,113,007,153,212
 3111 :114,007,153,153,007,153,114
 3117 :154,007,152,010,010,010,132
 3123 :072,144,010,169,002,013,205
 3129 :016,208,141,016,208,208,086
 3135 :008,169,253,045,016,208,250
 3141 :141,016,208,104,024,105,155
 3147 :028,141,002,208,169,226,081
 3153 :141,003,208,169,247,141,222
 3159 :249,007,169,171,141,037,093
 3165 :017,173,031,208,044,017,071
 3171 :208,048,251,173,018,208,237
 3177 :201,242,208,244,169,007,152
 3183 :141,040,208,169,002,013,172
 3189 :021,208,141,021,208,173,121
 3195 :031,208,024,096,072,138,180
 3201 :072,162,005,254,200,007,061
 3207 :169,058,221,200,007,208,230
 3213 :008,169,048,157,200,007,218
 3219 :202,208,238,104,170,104,149
 3225 :096,072,138,072,173,205,141
 3231 :007,024,105,005,201,058,047
 3237 :176,005,141,205,007,208,139
 3243 :234,233,010,141,205,007,233
 3249 :162,004,208,207,072,138,200
 3255 :072,162,004,208,200,120,181
 3261 :169,049,141,020,003,169,228
 3267 :234,141,021,003,088,169,083
 3273 :048,141,214,007,032,132,007
 3279 :255,160,016,185,046,013,114
 3285 :201,064,144,003,056,233,146
 3291 :064,153,011,004,169,003,111
 3297 :153,011,216,136,208,235,160
 3303 :200,140,026,008,173,026,036
 3309 :008,208,012,169,045,141,052
 3315 :025,004,169,062,141,026,158
 3321 :004,208,010,169,060,141,073
 3327 :025,004,169,045,141,026,153
 3333 :004,173,000,220,045,001,192
 3339 :220,074,074,074,176,005,122
 3345 :160,001,140,026,008,074,170
 3351 :176,005,160,000,140,026,018
 3357 :008,074,176,202,032,129,138
 3363 :255,173,026,008,240,003,228
 3369 :076,027,008,133,198,000,227
 3375 :080,076,065,089,032,065,198
 3381 :071,065,073,078,063,032,179
 3387 :089,032,032,078,169,128,075
 3393 :141,011,212,169,129,141,100
 3399 :011,212,096,169,255,056,102
 3405 :237,003,208,141,015,212,125
 3411 :096,169,000,141,015,212,204
 3417 :096,169,024,141,025,008,040
 3423 :165,162,141,024,008,169,252
 3429 :064,141,004,212,169,065,244

LOAD

Heat Seeker



3435 :141,004,212,096,165,162,119
 3441 :205,024,008,240,016,141,235
 3447 :024,008,173,025,008,201,046
 3453 :048,176,007,141,001,212,198
 3459 :238,025,008,096,169,000,155
 3465 :141,001,212,096,076,153,048
 3471 :013,011,000,000,000,000,167
 3477 :000,000,000,000,120,169,182
 3483 :171,141,020,003,169,013,160
 3489 :141,021,003,169,014,141,138
 3495 :005,220,088,096,173,000,237
 3501 :220,045,001,220,141,146,178
 3507 :013,173,034,017,013,035,208
 3513 :017,208,003,076,195,014,186
 3519 :173,146,013,041,016,208,020
 3525 :003,238,066,017,173,146,072
 3531 :013,041,003,201,003,208,160
 3537 :003,076,195,014,173,144,046
 3543 :013,141,145,013,173,034,222
 3549 :017,141,149,013,048,003,080
 3555 :169,000,044,169,255,141,237
 3561 :150,013,160,008,136,014,202
 3567 :145,013,144,250,192,000,215
 3573 :240,037,014,149,013,046,232
 3579 :150,013,014,145,013,144,218
 3585 :023,024,173,149,013,109,236
 3591 :034,017,141,149,013,144,249
 3597 :003,238,150,013,173,034,112
 3603 :017,016,003,206,150,013,168
 3609 :136,208,219,173,146,013,152
 3615 :041,002,208,022,056,173,021
 3621 :148,013,237,149,013,141,226
 3627 :148,013,173,035,017,237,154
 3633 :150,013,141,035,017,076,225
 3639 :076,014,024,173,148,013,247
 3645 :109,149,013,141,148,013,122
 3651 :173,035,017,109,150,013,052
 3657 :141,035,017,173,144,013,084
 3663 :141,145,013,173,035,017,091
 3669 :141,151,013,048,003,169,098
 3675 :000,044,169,255,141,152,084
 3681 :013,160,008,136,014,145,061
 3687 :013,144,250,192,000,240,174
 3693 :037,014,151,013,046,152,010
 3699 :013,014,145,013,144,023,211
 3705 :024,173,151,013,109,035,114
 3711 :017,141,151,013,144,003,084
 3717 :238,152,013,173,035,017,249
 3723 :016,003,206,152,013,136,153

3729	:208,219,173,146,013,041,177	4071	:077,037,017,048,212,016,126
3735	:002,208,022,024,173,147,215	4077	:023,173,215,014,056,077,027
3741	:013,109,151,013,141,147,219	4083	:037,017,048,011,172,210,226
3747	:013,173,034,017,109,152,149	4089	:014,204,211,014,208,003,135
3753	:013,141,034,017,076,195,133	4095	:076,012,017,169,000,044,061
3759	:014,056,173,147,013,237,047	4101	:169,001,044,212,014,016,205
3765	:151,013,141,147,013,173,051	4107	:002,073,001,044,213,014,102
3771	:034,017,237,152,013,141,013	4113	:016,002,073,001,172,210,235
3777	:034,017,076,198,014,076,096	4119	:014,204,211,014,240,005,199
3783	:216,014,008,000,000,000,181	4125	:073,001,141,203,014,173,122
3789	:000,000,000,000,000,000,205	4131	:201,014,141,202,014,173,012
3795	:000,000,000,000,000,173,128	4137	:036,017,141,206,014,048,247
3801	:000,208,056,237,002,208,160	4143	:003,169,000,044,169,255,175
3807	:144,002,024,036,056,106,079	4149	:141,207,014,160,008,136,207
3813	:141,212,014,173,001,208,210	4155	:014,202,014,144,250,192,107
3819	:056,237,003,208,144,002,117	4161	:000,240,037,014,206,014,064
3825	:024,036,056,106,141,213,049	4167	:046,207,014,014,202,014,056
3831	:014,169,003,045,016,208,190	4173	:144,023,024,173,206,014,149
3837	:240,046,201,003,240,042,001	4179	:109,036,017,141,206,014,094
3843	:201,001,240,019,173,212,081	4185	:144,003,238,207,014,173,100
3849	:014,056,233,128,201,165,038	4191	:036,017,016,003,206,207,068
3855	:176,003,024,105,172,141,124	4197	:014,136,208,219,173,203,030
3861	:212,014,076,069,015,173,068	4203	:014,208,022,056,173,205,017
3867	:212,014,024,105,128,201,199	4209	:014,237,206,014,141,205,162
3873	:085,144,003,056,233,172,214	4215	:014,173,037,017,237,207,036
3879	:141,212,014,076,069,015,054	4221	:014,141,037,017,076,151,049
3885	:173,212,014,048,009,201,190	4227	:016,024,173,205,014,109,160
3891	:085,048,012,056,233,172,145	4233	:206,014,141,205,014,173,122
3897	:144,007,201,165,016,003,081	4239	:037,017,109,207,014,141,156
3903	:024,105,172,141,212,014,219	4245	:037,017,173,201,014,141,220
3909	:173,084,017,048,003,076,214	4251	:202,014,173,037,017,141,227
3915	:012,017,169,000,044,037,098	4257	:208,014,048,003,169,000,091
3921	:017,048,009,044,036,017,252	4263	:044,169,255,141,209,014,231
3927	:016,014,169,001,208,010,249	4269	:160,008,136,014,202,014,195
3933	:044,036,017,016,003,169,122	4275	:144,250,192,000,240,037,018
3939	:002,044,169,003,141,210,156	4281	:014,208,014,046,209,014,178
3945	:014,169,000,044,213,014,047	4287	:014,202,014,144,023,024,100
3951	:048,009,044,212,014,016,198	4293	:173,208,014,109,037,017,243
3957	:014,169,001,208,010,044,051	4299	:141,208,014,144,003,238,183
3963	:212,014,016,003,169,002,027	4305	:209,014,173,037,017,016,163
3969	:044,169,003,141,211,014,199	4311	:003,206,209,014,136,208,223
3975	:173,210,014,056,237,211,012	4317	:219,173,203,014,208,022,036
3981	:014,074,144,008,041,001,167	4323	:024,173,204,014,109,208,191
3987	:141,203,014,076,034,016,119	4329	:014,141,204,014,173,036,047
3993	:240,018,169,000,056,237,105	4335	:017,109,209,014,141,036,253
3999	:212,014,141,212,014,169,153	4341	:017,076,012,017,056,173,084
4005	:000,056,237,213,014,141,058	4347	:204,014,237,208,014,141,045
4011	:213,014,169,000,141,214,154	4353	:204,014,173,036,017,237,170
4017	:014,141,215,014,173,212,178	4359	:209,014,141,036,017,076,244
4023	:014,013,213,014,208,003,136	4365	:015,017,076,108,017,000,246
4029	:076,012,017,173,214,014,183	4371	:000,000,000,000,000,000,019
4035	:024,109,212,014,141,214,141	4377	:000,000,000,000,000,000,025
4041	:014,173,215,014,024,109,238	4383	:000,000,000,000,000,000,031
4047	:213,014,141,215,014,173,209	4389	:000,000,000,000,000,000,037
4053	:214,014,056,237,036,017,019	4395	:000,000,000,000,000,000,043
4059	:077,036,017,016,014,173,040	4401	:000,000,000,000,000,000,049
4065	:215,014,056,237,037,017,033	4407	:000,000,000,000,000,000,055

4413 :000,000,000,000,000,000,061
 4419 :000,000,000,000,000,000,067
 4425 :000,000,000,000,000,000,073
 4431 :000,000,000,000,000,000,079
 4437 :001,254,002,253,004,251,082
 4443 :008,247,016,239,032,223,088
 4449 :064,191,128,127,000,000,095
 4455 :000,000,000,000,000,162,009
 4461 :000,189,050,017,141,101,095
 4467 :017,189,034,017,141,102,103
 4473 :017,048,003,169,000,044,146
 4479 :169,255,141,103,017,189,233
 4485 :035,017,141,104,017,048,239
 4491 :003,169,000,044,169,255,011
 4497 :141,105,017,160,008,136,200
 4503 :014,101,017,144,250,192,101
 4509 :000,240,066,014,102,017,084
 4515 :046,103,017,014,104,017,208
 4521 :046,105,017,014,101,017,213
 4527 :144,046,024,173,102,017,169
 4533 :125,034,017,141,102,017,105
 4539 :144,003,238,103,017,189,113
 4545 :034,017,016,003,206,103,060
 4551 :017,024,173,104,017,125,147
 4557 :035,017,141,104,017,144,151
 4563 :003,238,105,017,189,035,030
 4569 :017,016,003,206,105,017,069
 4575 :136,208,190,169,000,141,043
 4581 :106,017,189,085,017,045,176
 4587 :016,208,240,003,238,106,022
 4593 :017,024,189,018,017,109,103
 4599 :102,017,157,018,017,189,235
 4605 :000,208,109,103,017,157,079
 4611 :000,208,144,003,238,106,190
 4617 :017,044,103,017,016,003,209
 4623 :206,106,017,044,106,017,255
 4629 :016,014,169,001,141,106,212
 4635 :017,024,189,000,208,105,058
 4641 :088,157,000,208,169,001,144
 4647 :205,106,017,208,019,189,015
 4653 :000,208,201,088,144,012,186
 4659 :206,106,017,056,189,000,113
 4665 :208,233,088,157,000,208,183
 4671 :173,106,017,208,012,189,000
 4677 :086,017,045,016,208,141,070
 4683 :016,208,076,089,018,189,159
 4689 :085,017,013,016,208,141,049
 4695 :016,208,024,189,019,017,048
 4701 :109,104,017,157,019,017,004
 4707 :189,001,208,109,105,017,216
 4713 :201,029,176,002,169,029,199
 4719 :201,250,144,002,169,250,103
 4725 :157,001,208,232,232,224,147
 4731 :016,240,003,076,110,017,073
 4737 :044,067,017,048,003,238,034
 4743 :067,017,162,005,254,068,196
 4749 :017,202,016,250,173,084,115



4755	:017,048,003,238,084,017,042
4761	:173,036,017,208,013,173,005
4767	:037,017,208,008,169,255,085
4773	:141,249,007,076,001,019,146
4779	:169,000,141,107,017,173,010
4785	:036,017,048,012,201,032,011
4791	:144,017,169,004,141,107,253
4797	:017,076,202,018,201,224,159
4803	:176,005,169,006,141,107,031
4809	:017,173,037,017,048,016,253
4815	:201,032,144,025,024,169,034
4821	:008,109,107,017,141,107,190
4827	:017,076,236,018,201,224,223
4833	:176,009,024,169,009,109,209
4839	:107,017,141,107,017,173,025
4845	:107,017,201,004,208,002,008
4851	:169,010,201,006,208,002,071
4857	:169,011,024,105,238,141,169
4863	:249,007,173,034,017,208,175
4869	:013,173,035,017,208,008,203
4875	:169,255,141,248,007,076,139
4881	:105,019,169,000,141,107,046
4887	:017,173,034,017,048,012,068
4893	:201,035,144,017,169,004,087
4899	:141,107,017,076,050,019,189
4905	:201,224,176,005,169,006,054
4911	:141,107,017,173,035,017,025
4917	:048,016,201,032,144,025,007
4923	:024,169,008,109,107,017,237
4929	:141,107,017,076,084,019,253
4935	:201,224,176,009,024,169,106
4941	:009,109,107,017,141,107,055
4947	:017,173,107,017,201,004,090
4953	:208,002,169,010,201,006,173
4959	:208,002,169,011,024,105,102
4965	:230,141,248,007,076,049,084
4971	:234,076,114,019,076,219,077
4977	:020,032,181,255,120,173,126
4983	:022,208,009,016,141,022,025
4989	:208,169,029,141,024,208,136
4995	:169,007,141,035,208,169,092
5001	:000,141,037,208,169,007,187
5007	:141,038,208,169,015,141,087
5013	:039,208,169,147,133,254,075
5019	:169,022,133,255,169,128,007
5025	:133,252,169,059,133,253,136
5031	:160,000,177,254,208,024,222
5037	:230,254,208,002,230,255,072

5043	:177,254,170,169,000,145,070	5385	:020,010,170,189,231,004,121
5049	:252,230,252,208,002,230,079	5391	:073,128,157,231,004,032,128
5055	:253,202,208,243,240,008,065	5397	:086,022,240,030,189,231,051
5061	:145,252,230,252,208,002,006	5403	:004,073,128,157,231,004,112
5067	:230,253,230,254,208,002,100	5409	:152,024,109,247,020,201,018
5073	:230,255,165,252,201,064,096	5415	:255,208,002,169,000,201,106
5079	:208,208,169,254,141,028,199	5421	:003,208,002,169,001,141,057
5085	:208,165,001,041,251,133,252	5427	:247,020,208,208,173,247,130
5091	:001,160,000,185,000,220,025	5433	:020,024,105,004,141,050,145
5097	:153,000,048,185,000,221,072	5439	:017,105,002,141,052,017,141
5103	:153,000,049,200,208,241,066	5445	:105,003,141,201,014,105,126
5109	:165,001,009,004,133,001,046	5451	:001,141,144,013,169,002,033
5115	:160,000,185,154,020,153,155	5457	:205,247,020,208,003,238,234
5121	:000,050,200,192,032,208,171	5463	:144,013,169,255,141,248,033
5127	:245,160,000,152,153,000,205	5469	:020,160,025,162,026,169,143
5133	:051,200,192,008,208,248,152	5475	:022,032,073,022,173,248,157
5139	:169,004,133,255,169,216,197	5481	:020,010,170,232,232,189,190
5145	:133,253,169,000,133,254,199	5487	:044,005,073,128,157,044,050
5151	:133,252,168,169,096,145,226	5493	:005,032,086,022,240,035,025
5157	:254,169,008,145,252,200,041	5499	:024,109,248,020,189,044,245
5163	:208,245,230,255,230,253,184	5505	:005,073,128,157,044,005,029
5169	:165,255,201,007,208,235,096	5511	:152,024,109,248,020,201,121
5175	:169,096,153,000,007,169,137	5517	:254,208,002,169,255,201,206
5181	:008,153,000,219,200,192,065	5523	:001,208,002,169,000,141,156
5187	:192,208,241,032,219,020,211	5529	:248,020,076,103,021,160,013
5193	:160,000,169,254,153,250,035	5535	:021,162,051,169,022,032,104
5199	:007,169,002,153,041,208,147	5541	:073,022,032,086,022,208,096
5205	:200,192,006,208,241,169,077	5547	:251,096,197,032,201,032,212
5211	:255,141,248,007,141,249,108	5553	:206,032,058,084,082,069,196
5217	:007,160,000,185,186,020,143	5559	:080,088,069,032,044,069,053
5223	:201,064,144,003,056,233,036	5565	:084,065,073,068,069,077,113
5229	:064,153,192,007,169,003,185	5571	:082,069,084,078,073,032,101
5235	:153,192,219,200,044,248,147	5577	:044,069,067,073,086,079,107
5241	:020,016,006,192,024,208,075	5583	:078,032,017,013,045,084,220
5247	:228,240,004,192,033,208,008	5589	:067,069,076,069,083,032,097
5253	:222,169,003,153,192,219,067	5595	:079,084,032,078,079,084,143
5259	:200,192,040,208,248,088,091	5601	:084,085,066,032,068,078,126
5265	:096,096,066,064,096,096,147	5607	:065,032,075,067,073,084,115
5271	:067,065,096,000,000,000,123	5613	:083,089,079,074,032,069,151
5277	:000,000,003,003,003,003,169	5619	:083,085,032,017,013,053,014
5283	:003,011,011,043,043,040,058	5625	:056,057,049,032,033,069,033
5289	:040,000,000,000,000,000,209	5631	:084,085,080,077,079,195,087
5295	:192,192,192,192,192,224,079	5637	:032,044,210,197,203,197,120
5301	:224,232,232,040,040,080,005	5643	:197,211,032,212,193,197,029
5307	:076,065,089,069,082,049,105	5649	:200,032,032,032,032,032,121
5313	:058,032,032,032,032,032,155	5655	:155,017,147,014,050,032,182
5319	:032,032,032,083,072,073,011	5661	:049,032,058,083,082,069,146
5325	:080,083,058,032,032,032,010	5667	:089,065,076,080,032,070,191
5331	:080,076,065,089,069,082,160	5673	:079,032,082,069,066,077,190
5337	:050,058,160,000,162,004,139	5679	:085,078,032,017,013,078,094
5343	:189,145,020,153,112,007,081	5685	:073,071,069,066,032,079,187
5349	:189,149,020,153,152,007,131	5691	:084,032,069,082,073,070,213
5355	:200,202,208,240,192,040,037	5697	:032,083,083,069,082,080,238
5361	:208,234,096,076,249,020,100	5703	:017,013,134,254,133,255,109
5367	:000,000,169,000,141,247,036	5709	:177,254,032,210,255,136,117
5373	:020,160,110,162,172,169,022	5715	:208,248,096,173,000,220,004
5379	:021,032,073,022,173,247,059	5721	:045,001,220,041,028,201,113



5727 :028,208,244,169,000,133,109
 5733 :162,169,028,197,162,208,003
 5739 :252,173,000,220,045,001,030
 5745 :220,041,004,208,003,160,237
 5751 :255,096,173,000,220,045,140
 5757 :001,220,041,008,208,003,094
 5763 :160,001,096,173,000,220,013
 5769 :045,001,220,041,016,208,156
 5775 :220,160,000,096,000,008,115
 5781 :192,000,001,027,192,000,049
 5787 :001,063,192,000,001,063,219
 5793 :128,000,001,062,000,002,098
 5799 :060,000,002,060,000,002,035
 5805 :060,000,002,062,000,002,043
 5811 :063,000,002,063,000,002,053
 5817 :063,000,002,062,000,002,058
 5823 :062,000,002,060,000,002,061
 5829 :060,000,002,056,000,002,061
 5835 :048,000,009,012,000,002,018
 5841 :028,000,002,060,000,002,045
 5847 :060,000,002,124,000,002,147
 5853 :124,000,002,252,000,002,089
 5859 :252,000,002,252,000,002,223
 5865 :124,000,002,060,000,002,165
 5871 :060,000,002,060,000,002,107
 5877 :124,000,001,001,252,000,111
 5883 :001,003,252,000,001,003,255
 5889 :216,000,001,003,000,027,248
 5895 :224,000,002,248,000,002,227
 5901 :126,003,224,063,255,252,168
 5907 :127,255,255,063,255,255,205
 5913 :000,055,255,255,252,255,073
 5919 :255,254,063,255,252,007,093
 5925 :192,126,000,002,031,000,132
 5931 :002,007,000,022,003,000,077
 5937 :002,007,000,002,015,000,075
 5943 :002,015,000,002,063,000,137
 5949 :002,063,128,000,001,015,014
 5955 :224,000,001,003,248,000,031
 5961 :002,255,128,000,001,063,010
 5967 :224,000,001,015,224,000,031
 5973 :001,003,248,000,002,248,075
 5979 :000,002,060,000,002,012,167
 5985 :000,032,003,252,000,001,129
 5991 :015,252,000,001,031,240,130
 5997 :000,001,031,192,224,127,172
 6003 :000,001,249,252,000,001,106
 6009 :127,240,000,001,127,192,040
 6015 :000,001,063,000,002,060,253

6021 :000,002,016,000,035,008,194
 6027 :000,002,060,000,002,252,199
 6033 :000,001,003,254,000,001,148
 6039 :015,254,000,001,063,159,131
 6045 :000,001,254,007,003,248,158
 6051 :000,001,015,248,000,001,172
 6057 :063,240,000,001,063,192,216
 6063 :000,032,048,000,002,060,061
 6069 :000,002,031,000,002,031,247
 6075 :192,000,001,007,240,000,115
 6081 :001,007,252,000,001,001,199
 6087 :255,000,002,031,192,000,167
 6093 :001,007,240,000,001,001,199
 6099 :252,000,002,252,000,002,207
 6105 :240,000,002,240,000,002,189
 6111 :224,000,002,192,000,019,148
 6117 :040,040,000,001,040,040,134
 6123 :000,001,041,104,000,001,126
 6129 :041,104,000,001,009,096,236
 6135 :000,001,009,096,000,001,098
 6141 :001,064,000,001,001,064,128
 6147 :000,001,001,064,000,001,070
 6153 :001,064,000,001,001,064,140
 6159 :000,032,001,064,000,001,113
 6165 :001,064,000,001,001,064,152
 6171 :000,001,001,064,000,001,094
 6177 :001,064,000,001,009,096,204
 6183 :000,001,009,096,000,001,146
 6189 :041,104,000,001,041,104,080
 6195 :000,001,040,040,000,001,133
 6201 :040,040,000,035,170,000,086
 6207 :002,170,128,000,001,021,129
 6213 :085,000,001,021,085,000,005
 6219 :001,021,085,000,001,170,097
 6225 :128,000,001,170,000,046,170
 6231 :170,000,001,002,170,000,174
 6237 :001,085,084,000,001,085,093
 6243 :084,000,001,085,084,000,097
 6249 :001,002,170,000,002,170,194
 6255 :000,044,008,000,002,010,175
 6261 :000,002,006,128,000,001,254
 6267 :021,128,000,001,165,064,246
 6273 :000,001,041,080,000,001,252
 6279 :010,084,000,002,021,000,252
 6285 :002,005,000,039,005,000,192
 6291 :002,021,000,001,010,084,009
 6297 :000,001,041,080,000,001,020
 6303 :165,064,000,001,021,128,026
 6309 :000,001,006,128,000,001,045
 6315 :010,000,002,008,000,040,231
 6321 :032,000,002,160,000,001,116
 6327 :002,144,000,001,002,084,160
 6333 :000,001,001,090,000,001,026
 6339 :005,104,000,001,021,160,230
 6345 :000,001,084,000,002,080,112
 6351 :000,039,080,000,002,084,156
 6357 :000,002,021,160,000,001,141

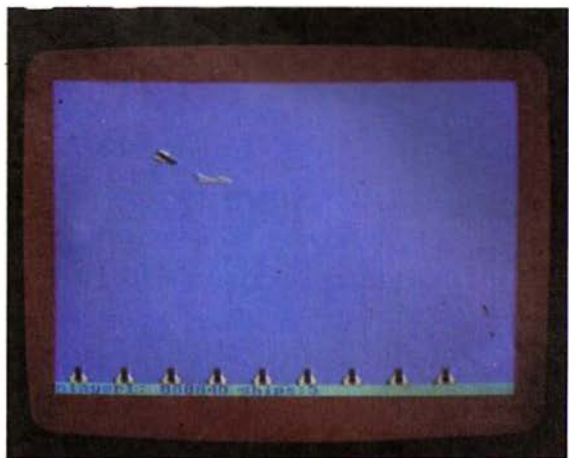
LOAD



Heat Seeker

6453 :160,042,255,224,047,255,012
6459 :248,043,187,224,010,170,173
6465 :168,255,013,013,013,130,145

6363 :005,104,000,001,001,090,164
6369 :000,001,002,084,000,001,057
6375 :002,144,000,002,160,000,027
6381 :002,032,000,026,008,128,177
6387 :000,001,010,168,000,001,167
6393 :043,224,000,001,011,224,240
6399 :000,001,011,232,000,001,244
6405 :042,160,000,001,002,032,242
6411 :000,051,136,000,001,002,201
6417 :170,000,001,002,174,000,108
6423 :001,002,238,128,010,255,145
6429 :160,010,255,224,011,254,175
6435 :168,011,255,224,042,255,222
6441 :168,043,255,224,043,255,005
6447 :232,011,255,224,047,255,047



Novità Jackson.

CON
CASSETTA



GRUPPO
EDITORIALE
JACKSON



La biblioteca che fa testo.

MISSIONE AFRICA

**Una "Nave della Pace" in partenza dall'Italia
porterà in Africa soccorsi immediati e aiuti
per un domani migliore con il vostro contributo.**



La "Nave della Pace": uno strumento nuovo per intervenire immediatamente là dove è necessario un urgente soccorso a popolazioni gravemente colpite da calamità naturali. Non porta solo aiuti immediati ma anche strumenti di lavoro e mezzi ausiliari idonei a realizzare migliori condizioni di vita.

Affinché questo risultato sia raggiunto, gli aiuti siano adeguati ed efficaci e la "Nave della Pace" possa partire a pieno carico occorrono offerte e solidarietà da parte di tutti. Specialisti dei pro-

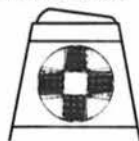
blemi dello sviluppo hanno offerto consigli per la migliore attuazione del progetto. L'iniziativa ha ottenuto l'Alto Patronato del Presidente della Repubblica Italiana, il



Patrocinio dell'ONU, della Lega di Società di Croce Rossa e di Mezza Luna Rossa e della Croce Rossa Italiana e il

contributo del Dipartimento per la Cooperazione allo Sviluppo del Ministero degli Affari Esteri. Il Comitato "Nave della Pace" chiede a quanti comprendono la necessità ed il valore dell'iniziativa, offerte di

mezzi e di beni indispensabili per i soccorsi di prima necessità e per gli aiuti di sviluppo destinati a 15 paesi dell'Africa.



Nave della Pace

COMITATO NAVE DELLA PACE - VIALE MAZZINI 41
00195 ROMA - TEL. 06/317447-308163

Da compilare in stampatello ed inviare in busta chiusa a:

Comitato Nave della Pace
V.le Mazzini 41 - 00195 Roma.

**SI', ANCH'IO VOGLIO AIUTARE
LA NAVE DELLA PACE A PARTIRE
CON LE STIVE PIENE**

Cognome _____

Nome _____

Via _____ N. _____

CAP _____ Località _____

Per questo ho deciso di inviare il mio contributo di

☐ Lit. 10.000 ☐ Lit. 50.000
☐ Lit. 25.000 ☐ Lit. 100.000 o più

tramite:

☐ Assegno non trasferibile intestato:

Comitato Nave della Pace

☐ C/c postale n. 15285000

☐ Bonifico bancario a credito

del c/c n. 3100/51 c/o

la Cassa di Risparmio di Roma

sede centrale - Via del Corso, 320 - 00186 Roma

☐ Desidero una ricevuta del mio versamento.

VIC-sound

ADSR: con il VIC il suono del C64

di V. Silvestroni

Ecco a voi un programma in BASIC, che simula l'AttackDecay-SustainRelease del C64 e "gira" su VIC 20 con espansione di memoria da 3 Kbyte.

Come funzionano questi parametri in una normale tastiera musicale, così il volume (loc. 36878) del VIC 20 si impegna a crearne una verosimile imitazione.

Il tutto coi limiti di esecuzione del BASIC che però sono stati ridotti all'osso. Andiamo per ordine. Innanzitutto poche DATA, che servono per le note e per i valori di default. Poi lo schermo vi mostra la PRIMA VIDEATA.

ADSR in alto, subito sotto la scritta LEVEL (livello), poi, ancora sotto, ATT. DEC e SUSTAIN e sotto queste parole due numeri (default 15 e 5), quindi la parola TIME (tempo), poi ATT. DEC. SUS. e sotto i valori (default 1,3 e 8). Ancora sotto i segni "+" e "-" ed infine la scritta "C=Ctrl (RET) =avanti".

Come comandare l'involuppo.

Sotto alcune delle parole sopracitate stanno scritte le abbreviazioni usate per indicare quanti e quali tasti premere per cambiare un certo parametro. Quindi Shift=ATT.DEC, Cmdr=SUSTAIN, Ctrl=ATT., Shift+Ctrl=DEC., Cmdr+Ctrl=SUS.. Premendo solo questi tasti non cambia nulla, ma se tenete premuto, ad esempio, SHIFT e contemporaneamente il tasto "-", vedrete che il numero sotto ATT.DEC. decresce fino a 0. 0 è il minimo che tutti i parametri accettano. Premendo SHIFT e "+" vedrete che il numero sale fino a 15, per poi fermarsi.

15 è il massimo che i parametri ATT. DEC e SUSTAIN accettano (255 è il massimo per gli altri parametri). Questi numeri sottintendono il livello massimo che viene raggiunto dall'Attacco e tenuto dal Sustain (Sostegno). Gli altri valori sono il tempo (in unità Jiffy Clock=1 sessantesimo di secondo)



che il volume deve impiegare per raggiungere un determinato valore a passi di 1: cioè, se ATT.= 1, allora il volume impiega tanti sessantesimi (almeno) di secondo quante sono le unità comprese nel parametro ATT.DEC. per andare da 0 ad ATT.DEC.

Se ATT. o DEC. o SUS. sono uguali a 0 (anche non tutti insieme), il programma provvede, tramite variazioni sulle variabili, a rendere più veloci i tempi di esecuzione.

SECONDA VIDEATA. Gli effetti. È stato usato in questa sede il generatore di rumore bianco per ottenere un "similia"

di alcuni effetti (come il vibrato) e inventarne altri (come il vento). Cioè sono stati simulati vari effetti tramite la digitazione di note stonate tra loro. Qui voi dovete scegliere un effetto fra 5 (che comunque potrete escludere in esecuzione col tasto CTRL).

INFINE. Viene mostrata una videata contenente in alto i valori da voi scelti nella prima videata con i controlli a disposizione (shift-cmdr-ctrl-"+"-"), i generatori accesi/spenti (bianco o niente = spento; numero = acceso), i tasti da usare per suonare e cioè (da Dodie-sis a Do diesis per un'estensione di 2 ot-

tave):

12 456 89 + - £ins/del
QWERTYUIOP @ + !clr/home
Sotto viene scritto il nome dell'effetto
scelto e un promemoria per i vari con-
trolli di questa pagina.

Il programma è interamente in BASIC
e fa uso esclusivo, per suonare e cam-
biare suono, dei comandi e funzioni PO-
KE, PEEK e di variazioni e operazioni
tra variabili.

Per tornare alla pagina delle variabili
dalla pagina della musica premere con-
temporaneamente SHIFT e CTRL.

Le parti essenziali del listato sono due:
quella delle "DECISIONI" parte dalla
linea 100 e finisce alla linea 235, quella
del suono vero e proprio parte dalla
linea 390 per arrivare alla 670. In que-
sti due blocchi di linee è racchiuso il
buon funzionamento del programma; le
altre servono in massima parte per pre-
sentare il programma.

Buon divertimento.

P.S. Mettendo la variabile ATT.DEC a
0, o comunque sistemando le variabili
in modo che ATT.DEC sia minore di
SUSTAIN, avrete delle variazioni nel-

l'involuppo. Queste variazioni possono
sia risultare indesiderate che utilizzabili
altrimenti. In pratica alcune volte il vo-
lume può andare a zero (0) senza passa-
re per il release oppure fermarsi a zero
durante il decay. Questo "fastidio" si
può eliminare mettendo un'opportuna
linea tra la 200 e la 245 del tipo: 225
IF AD < SL THEN AD=SL.

Abbiamo volutamente lasciato l'inghip-
po, perché, volendo, si possono sfrutta-
re queste sue "variazioni nell'involuppo"
per creare musica. Viva gli involuppi
pseudo-casuali.

ADSR per VIC 20

```

5 PRINTCHR$(147);CHR$(8);CHR$(5);:POKE368
79,110:L=8164:GOSUB4000:rem 190
10 DIMADSR(6),S(4),NE%(4),ES(4),NO%(64),C
O%(6),RI%(6):V=36878:V%=0:rem 248
20 FORT=1TO4:S(T)=36873+T:NEXT:FORT=1TO6:
READADSR(T),CO%(T),RI%(T):NEXT
:rem 199
30 FORT=0TO15:READNO%(T),NO%(T+48):NO%(T+
16)=0:NO%(T+32)=0:NEXT:FORT=0TO4:READN
E%(T):NEXT:rem 151
40 NO%(64)=0:FORT=0TO4:READES(T):NEXT
:rem 228
50 POKE36869,242:PRINTCHR$(19)SPC(9)CHR$(
18)"ADSR":PRINT:PRINTSPC(8)CHR$(18)"L
EVEL":rem 166
53 POKEV,0:rem 73
55 PRINT:PRINT"A{ 2 T}:DEC"SPC(8)"SUSTAI
N":PRINT"SHIFT"SPC(13)"CMDR":rem 20
60 PRINTSPC(6)STR$(ADSR(1))TAB(12)STR$(AD
SR(2)):rem 32
63 PRINT:PRINT:PRINTSPC(9)CHR$(18)"TIME"
:rem 34
65 PRINT:PRINT"A{ 2 T}."SPC(5)"DEC."SPC(5)
)REL":PRINT"CTRL"SPC(5)"SH+C"SPC(5)
CM+C":rem 103
70 PRINTSTR$(ADSR(4))TAB(9)STR$(ADSR(5))T
AB(18)STR$(ADSR(6)):PRINT:rem 7
75 PRINTSPC(6)+"SPC(8)"-":PRINT:PRINT"C
=CTRL":SPC(4)"<RET>AVANTI":rem 206
100 A=PEEK(653):IFA=3ORA=7THEN100
:rem 179
110 PP=PEEK(197):IFA=0THEN180:rem 197
120 IFPP=5THENADSR(A)=ADSR(A)+1:rem 116
130 IFPP=61THENADSR(A)=ADSR(A)-1:rem 169
140 IFA=1ORA=2THENIFADSR(A)>15THENADSR(A)
=15:rem 208
150 IFA>3ORA<7THENIFADSR(A)>255THENADSR(A)
=255:rem 68
160 IFADSR(A)<0THENADSR(A)=0:rem 166
170 RI=RI%(A):CO=CO%(A):GOSUB5000:PRINTST
R$(ADSR(A)):":GOTO100:rem 7
180 :rem 211
190 :rem 212
200 IFPP>15THEN100:rem 97
210 AT=ADSR(4):DT=ADSR(5):RT=ADSR(6):SL=A
DSR(2):AD=ADSR(1):DV=1:DD=1:DR=1
:rem 188
220 IFAT=0THENDV=AD:rem 176
230 IFDT=0THENDD=SL:rem 188
235 IFSL=0ANDDT=0THENDD=AD:rem 134
240 IFRT=0THENDR=SL:rem 217
245 IFSL=0THENDR=AD:rem 189
247 PRINTCHR$(147);:L=7878:POKE36879,107:
POKE36867,PEEK(36867)AND129OR(20):GOS
UB4000:rem 154

```

```

250 PRINTSPC(9)CHR$(18)"ADSR":PRINT:PRINT
SPC(4)CHR$(18)"NOISE EFFETTO"
:rem 64
260 FORT=0TO4:PRINTES(T)SPC(8)STR$(T+1):N
EXT:rem 242
270 POKE198,0:WAIT198,1:GETAS:SC=VAL(AS)-
1:IFSC<0ORSC>4THEN270:rem 243
290 PRINTCHR$(147);:L=8164:POKE36879,110:
POKE36867,PEEK(36867)AND129OR4:GOSUB
4000:rem 62
300 PRINTCHR$(18)SPC(9)"ADSR":PRINT:PRINT
"AT:DE SUS. T.A T.D T.R":rem 164
303 ZS=STR$(AD):XS=STR$(SL):VS=STR$(AT):N
S=STR$(DT):MS=STR$(RT):POKE36869,240
:rem 94
305 PRINTCHR$(158)ZSSPC(6-LEN(ZS))XSSPC(4
-LEN(XS))VSSPC(4-LEN(VS))NSSPC(4-LEN(
NS))MS:rem 116
310 PRINTSPC(11)CHR$(5);CHR$(18);"GENERAT
ORI":rem 166
312 PRINT:rem 35
315 PRINT"12 456 89 + - £ CHR$(18)"T":PRIN
T"< 2 L> [< 3 L> [< 2 L>
[< 3 L> [< L>":PRINT"<M>{ 14 @}"
:rem 159
320 PRINT" QWERTYUIOPE*+CHR$(18)"S":PRIN
T" DEFGABCDEFABC":PRINT:rem 179
325 PRINTSPC(11)CHR$(18)"NOISE EFF. ":SPC
(11)ES(SC)SPC(1)"CONTROLLI":PRINTSPC
(29)"N.EF":rem 177
330 PRINTCHR$(146)"SHIFT CMDR SH+CM CTRL"
;SPC(2)"1"SPC(5)"2"SPC(4)"3"SPC(5)"4"
:rem 68
335 PRINT"BASSO ALTO TENOR":PRINT"CTRL+SH
IFT = MENU":rem 30
390 A=0:GOTO500:rem 82
400 IFPEEK(162)<RTTHEN500:rem 216
410 V%=V%-DR:IFV%<0THENV%=0:rem 82
420 POKEV,V%:POKE162,0:A=0:rem 8
500 S=PEEK(653):ONSGOTO700,700,700,700,75
0:rem 68
510 P=PEEK(197):IFP=64THEN400:rem 189
520 IFP=QTHEN590:rem 209
540 POKES(1),PEEK(S(1))AND128ORNO%(P):POK
ES(2),PEEK(S(2))AND128ORNO%(P)
:rem 175
550 POKES(3),PEEK(S(3))AND128ORNO%(P):POK
ES(4),PEEK(S(4))AND128ORNE%(SC)
:rem 244
560 Q=P:rem 121
590 ONAGOTO650:rem 80
600 IFPEEK(162)<ATTHEN500:rem 201
610 V%=V%+DV:IFV%>ADTHENA=1:V%=AD:GOTO500
:rem 243
620 POKEV,V%:GOTO500:rem 204
650 IFPEEK(162)<DTTHEN500:rem 209
660 V%=V%-DD:IFV%<SLTHENV%=SL:rem 41

```


VICsound

```

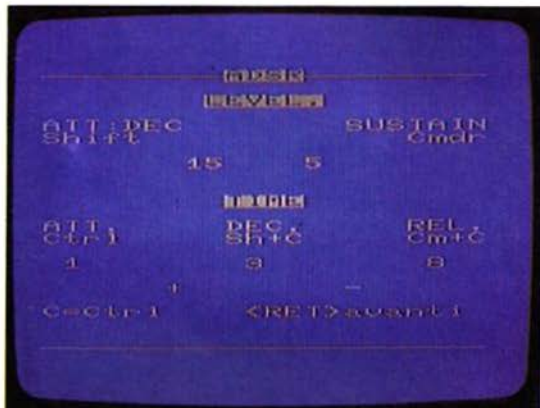
670 POKEV,V%:POKE162,0:GOTO500 :rem 47
700 IFSH(S)=0THENPOKES(S),PEEK(S(S))AND12
7:POKE7820+S*3,160:SH(S)=1:GOTO510
:rem 245
710 IFSH(S)=1THENPOKES(S),PEEK(S(S))OR128
:POKE7820+S*3,176+S:SH(S)=0:GOTO510
:rem 74
750 PRINTCHR$(147):GOSUB4000:PORT=1TO4:PO
KES(T),0:NEXT:GOTO50
:rem 93
3000 END :rem 154
4000 FORT=0TO21:POKE7680+T,64:POKE1+T,64:
NEXT:RETURN :rem 34
5000 POKE781,RI:POKE782,CO:SYS65520:RETUR
N :rem 120
6000 DATA15,6,7,5,12,7,...,1,15,3,9,15,8,
18,15 :rem 222
6100 DATA3,9,,28,43,49,,61,71,74,80,83,90
,92,96,,,15,22,33,39,53,57,65,68,,77
,85 :rem 163

```

```

6110 DATA88,94,,, :rem 67
6200 DATA127,126,125,124,123,"BRASS
{ 6 SPAZI}","VIBRATO{ 4 SPAZI}","TRE
MOLO{ 4 SPAZI}","VENTO{ 6 SPAZI}"
:rem 138
6210 DATA"SOFFIO{ 5 SPAZI}" :rem 237

```



LIBRI FIRMATI JACKSON

**CON
CASSETTA**

**COMMODORE
64**

i file

**COMMODORE
64**

il basic

**la grafica
e il suono**

LA BIBLIOTECA CHE FA TESTO.

P R I N T

di M. Cristub Grizzi

**Recensioni
Libri**

**I libri recensiti
possono essere
ordinati
utilizzando
il coupon
a pag. 98**

STATISTICA A UNA DIMENSIONE CON IL COMMODORE 64

di U. G. Barzaghi
Gruppo Editoriale Jackson



Crescendo la diffusione dell'home computer, cresce anche il numero di quelle persone, che, stanche di utilizzarlo unicamente per svago e divertimento, iniziano ad orientarsi verso applicazioni pratiche del proprio sistema.

Una delle applicazioni forse più "congeniali" ad un computer è proprio la statistica. Essa infatti, trattando una larga mole di dati e cercando in essi quelle correlazioni e similitudini che permettono di sintetizzare i risultati in un semplice grafico od una sola tabella, può essere velocemente ed affidabilmente gestita dal computer. Per ottenere buoni risultati occorre tuttavia conoscere almeno i fondamenti della statistica e scrivere un programma che permetta al calcolatore di eseguire tutte le operazioni necessarie sui dati che inseriamo.

Questo libro offre sia l'uno che l'altro, spiegando in modo didattico e di facile comprensione i principi basilari della statistica e fornendo in appendice un programma per C64 che vi metterà in grado di trattare in pochi secondi grandi masse di dati, sintetizzandone i risultati. Tra l'altro, il libro comprende anche una serie di esercizi statistici risolti con il calcolatore che permettono di comprendere più a fondo gli argomenti trattati.

Un volume quindi senz'altro didattico - gli argomenti trattati coprono circa la metà del primo capitolo di un qualsiasi testo universitario di statistica (disuguaglianza di Tchebycheff esclusa) - che contribuirà ad affascinare ed avvicinare a questa disciplina molti possessori di C64.

COMMODORE 64 A SCUOLA

di J. Scriven e P. Hall
Gruppo Editoriale Jackson

Se è vero che, prima o poi, ciò che accade negli Stati Uniti accadrà anche da noi, presto vedremo la comparsa massiccia dei computer tra i banchi di scuola.

Del resto la maggior parte degli utilizzatori di home computer è tutt'ora composta da ragazzi in età scolare, e non sarà quindi particolarmente traumatico passare dal quaderno a quadretti alla tastiera di un calcolatore.

Questo libro si rivolge a coloro i quali, con poca o nessuna esperienza di programmazione, desiderano imparare ad usare il C64, sviluppando nel contempo delle capacità di ragionamento logico-matematico: il C64 infatti, per sua natura, possiede capacità e potenzialità molto efficaci, che purtroppo non sono accessibili attraverso semplici istruzioni BASIC, ma solo con lunghe serie di PEEK e POKE. Il libro mostra come superare questi inconvenienti e come, una volta compreso qualcosa di più sul funzionamento del C64, si possa essere

in grado di scrivere subito interessanti programmi didattici.

Destinato a studenti, insegnanti e genitori, "Commodore 64 a scuola" è strutturato con i primi due capitoli che presentano programmi di supporto all'insegnamento di concetti matematici elementari; i capitoli seguenti, invece, sono dedicati ad applicazioni di tipo linguistico, dall'ortografia alla creazione di testi originali. Infine, l'ultimo capitolo mostra come sfruttare le tecniche di simulazione per spiegare concetti astratti, quali la probabilità, o per descrivere sistemi complessi come una chiusa od una centrale elettrica.

I listati contenuti sono di ottima leggibilità e contengono pochissimi caratteri di controllo, risultando così di facile comprensione e digitazione anche per i meno esperti.



**I libri vengono offerti
con il 10% di sconto
sul prezzo di copertina**

FINALMENTE!

La Softrivista che ti gasa!

**IL VERO GIOCO
COMINCIA ADESSO**

**IN EDICOLA
JACKSON SOFT
SERIE ORO**

I giochi esclusivi per
Commodore 64 e Spectrum 48 K
importati dall'Inghilterra, mai
presentati in Italia.
Una sfida Jackson al già visto, al
già fatto, al... già registrato.



La prima
puntata del
fantastico,
inedito
PYJAMARAMA



**QUALCOSA DI SUPER, DI INEDITO,
DI IRRESISTIBILE**

Corri in edicola, il vero gioco comincia solo adesso
e se sei davvero bravo partecipa alla "sfida al campione",
utilizzando il tagliando che troverai sull'ultima pagina
di copertina di ogni numero.



GRUPPO
EDITORIALE
JACKSON

input-output

Piccoli annunci

A Cologna Veneta, in provincia di Verona, si è formato un club del CBM 64 per scambi di programmi, esperienze, consigli. Disponibili ultime novità, manuali e oltre 1.000 programmi. Michele Danese - Via Gioacchino Rossini - 37044 Cologna Veneta (VR) - Tel. 0442/85287

Vendo, cambio software di tutti i tipi per C64. Moltissimi giochi di tutti i generi e per tutti i gusti solo su cassetta (L. 2.000 a programma). Scrivere o telefonare ore pasti. Alessandro Bagnasco - Via Boggio, 41 - 12100 Cuneo - Tel. 0171/62822

Compro, vendo, cambio giochi per CBM 64 e VIC 20. Dispongo anche di vari programmi utility per linguaggi macchina e di un libro. Per la lista dei giochi (L. 1.000 cadauno) telefonare oppure scrivere a: Giacomo Bergamini - Via Martiri della Libertà, 99 - 52010 Subbiano (AR) - Tel. 0575/48891

Vendo e cambio programmi per C64 di vario genere. Possiedo molti giochi in LM. Annuncio sempre valido. Ragazzi rivete, rispondo a tutti. Nicola Giulino - Viale Terracini, 63 - 96100 Siracusa - Tel. 0931/39939

Eccellente: per CBM 64 vendocassetta con 8 utility tra cui Simons' BASIC, word processor, Vocabolario, c/c a L. 30.000 più spese postali contrassegno. Spedisco inoltre 10 games chimici e spedisco 10 games. Alessandro Stasi - Via Dei Pelaghi, 156 - 57100 Livorno

Vendo a L. 50.000 cadauna cassetta con giochi (Pacman, Decathlon, Popeye, Donkey Kong, ecc.) ed utility (word processing, data base, ecc.) per C64. Inviare lettera con francobollo per risposta per ricevere lista. Roberto Gariboldi - Via Della Birona, 60 - 20052 Monza (MI)

Vendo VIC 20, con espansione regolabile 3-8-16 Kbyte, dotato di manuale ed alimentatore, cartidge Sargon II, alcune cassette di games, libro "Alla scoperta del VIC 20" a L. 350.000 trattabili. Scrivere o contattare telefonicamente. Felice Lanzetti - Via Trento, 4 Angone - 25040 Darfo Boario Terme (BS) - Tel. 0364/51001

Vendo corso completo "Pannelli Solari" a L. 320.000. Per CBM 64 vendo 300 programmi di utility (non originali) a L. 500.000; oppure 20 programmi, ognuno su cassetta, a L. 26.000. Su disco L. 28.000. Specificare. Spedizione contrassegno. Roberto Mangoni - Via B. Buozzi, 4 - 00049 Velletri (RM) - Tel. 06/963689

Compro, cambio, vendo programmi per il C64 e per il VIC 20. Per maggiori informazioni scrivere a: Massimo Tabasso - Piazza Molineris, 1 - 12038 Savigliano (CN)

Vendo collezione completa "Program" a L. 40.000. Pietro Masi - Via Scalo S. Lorenzo, 47 - 00185 Roma - Tel. 06/4955167

Vendo per VIC 20 o CBM 64 20 o più programmi in cassetta o disco scelti su una lista di circa 1.000 programmi. Scrivetemi, invierò a tutti una lista gratis (costo di una cassetta L. 6.000). Andrea Abbruciati - Via Tomaso Silani, 11 - 00143 Roma

Vendo programmi su disco o cassetta fra cui Decathlon, Hes Games, EasyScript; cambierei volentieri i miei programmi con Pet Speed. Simone Lombardi - Via Emanuele - 55041 Camaiore (LU) - Tel. 989728

Vendo circa 1.000 programmi sia nastro che su disco per C64. Richiedere la mia lista, prezzi veramente bassi ex Zanella - Condominio Ciclamino - 10090 Romano Cavenese (TO)

Vendo tutte le ultime novità per C64. Giochi, utility, gestionali, grafici, musicali, linguaggi su disco o nastro. Turbo Tape, Turbo Disk, Copy 190. Chiedere elenco. Stefano Barcaroli - Via B. Buozzi, 8 - 05021 Acquasparta (TR) - Tel. 0744/930387

Per la prima volta in Italia vendo programmi ottimi e nuovissimi per VIC 20 e C64. Per la lista inviare L. 500. Giovanni Hamam - Via G. D'Annunzio, 34 - 80053 Castellammare di Stabia (NA)

Occasionissima: vendo VIC 20 in buono stato a L. 120.000 trattabili. In più regalo 1 cartidge "Cosmic Cruncher" (Pacman) più 60 giochi su cassetta, tra cui: La Rana, Attacco Spaziale, Blitz, Attacco Alieno, il calcio. Telefonare dal 13.35-14. Salvatore Minardi - Via Roma, 187 - 95041 Caltagirone (CT) - Tel. 0933/21107

Vendo giochi, utility, copiatori per CBM 64 sia su nastro che su disco. Scrivere o telefonare ore 19-21. Francesco Basile - Via Ortigara, 14 - 33100 Treviso - Tel. 0422/579627

Si è costituito il Commodore Club "Valdichiana": con l'iscrizione di L. 30.000 si ha diritto ricevere periodicamente tutto il software disponibile a condizioni vantaggiose. Si effettuano acquisti, vendite e scambi programmi di ogni genere. Per informazioni e richieste della lista programmi scrivere (unendo L. 500 in francobollo) a: Commodore Club Valdichiana - Loc. Palazzolo, 2 - 53040 Rignano (SI) - Tel. 0577/663550

Vendo, cambio per C64 oltre 1.000 programmi, tutte le migliori novità a bassissimi prezzi. Tratto preferibilmente su disco. Per ricevere elenco telefonare o scrivere. Paolo Gallo - Via Garibaldi, 83 - 96014 Florida (SR) - Tel. 0931/941056

Vendo: VIC 20, "Introduzione al BASIC" parte 1 e 2 (in italiano), "Il libro del Commodore VIC 20", "Giochi per il vostro VIC 20", cartuccia "Jupiter Lander" e numerosi giochi su cassetta. Il tutto a sole L. 200.000. Telefonare ore pranzo. Roberto La Torre - Via Fiume, 10 - 70043 Monopoli (BA) - Tel. 080/748355

Vendo, cambio i migliori programmi per CBM 64: videogames, utility, gestionali. Richiedere la lista, prezzi interessanti. Disponibili circa 400 programmi sia su disco che su nastro. Giuseppe Di Lello - Corso Europa, 13 - 66054 Vasto (CH) - Tel. 4393

Compro C64 a prezzo da trattare. Vendo VIC 20 più espansione Vixen a prezzo da concordare. Allego numerosi programmi. Telefonare ore pasti o scrivere. Daniele Monciatti - Via Fonte Benedetta, 18 - 53100 Siena - Tel. 0577/47123

Vendo VIC 20 più registratore, 3 cartidge (Cosmic Cruncher, Raid On Fort Knox, Garden Wars), 40 giochi su cassetta a L. 200.000 Donati - Via A. Vivaldi, 18 - 21013 Gallarate (VA) - Tel. 799420

Vendo VIC 20 più registratore, 9 cassette giochi, paddles, cartuccia "Star Post" a L. 280.000. Ivana Rizzo - Piazza Diodoro Sicuro, 8 - 92100 Agrigento - Tel. 0922/21705

Cambio e vendo programmi per C64. Inviare le vostre liste o richieste. Dario Pausin - Via Pascoli, 24 - 34129 Trieste - Tel. 040/773766

Il Club 64 di Montagnana (provincia di Padova) recentemente costituitosi ha le migliori novità del software. Telefonare allo 0429/82469. Carlo Ferrari - Casella Postale 6 - 35044 Montagnana (PD) - Tel. 0429/82469

Vendo per CBM 64 centinaia di programmi di tutti i tipi e di alta qualità. Prezzo medio di un programma L. 3.000, 50 programmi a scelta L. 100.000. Per ricevere la lista scrivere o telefonare. Stefano Cannelli - Via Rivotorto, 303 - 06080 Assisi (PG) - Tel. 075/814112

Vendo programmi per CBM 64. Giochi originali a prezzi favolosi. Aspetto le vostre liste da tutta Italia. Ne ho più di 2.000, tutti originali e costruiti in inglese. Giuliano Peleggi - via G. Torrielli, 10 - 00151 Roma - Tel. 06/5280434

Compro software per C64 a prezzi molto bassi. Inviatemi le vostre liste. Programmi su cassetta. Rocco Coluccelli - Viale Ofanto, 118 - 71100 Foggia - Tel. 0881/76114

Favoloso nido programmi per VIC 20; a L. 30.000 una sestina di tre proposte: 1) 70 programmi in versione base; 2) 30 programmi a 3-8-16 Kbyte di espansione; 3) 40 programmi per 3 Kbyte super expander. A scelta si possono avere giochi ed utility. Luciano Baglioni - Via Della Verena, 20 - 00141 Roma - Tel. 06/8185710

Vendo programmi per C64. Tutti originali; ultime novità; arrivi settimanali dall'Inghilterra. Disponibili su disco o cassetta. Per informazioni rivolgersi a: Giuliano Peleggi - Via Torrielli, 10 - 00151 Roma - Tel. 06/5280434

Vendo giochi su cassetta per CBM 64 a L. 2.500 (Calcio, Pac Man, Pole Position, Donkey Kong, ecc.). Inoltre, ogni 4 programmi acquistati, uno in omaggio. Rivolgersi a: Daniel Citone - Via Della Balduina, 73 - 00136 Roma - Tel. 06/3452182

Cambio, vendo giochi ed utility di tutti i tipi. Prezzi modici. Spedire L. 300 per avere la lista oppure telefonare ore pasti. Andrea Santini - Viale Belio Rione Bolzano, 21 - 72100 Brindisi - Tel. 0831/471925

Vendo videogiochi per VIC 20, disponibili anche su cassetta. Inviare L. 1.000 per ricevere la lista. Piercarlo Onidi - Via Cavour, 72 - 10045 Piossasco (TO) - Tel. 011/9064948

Vendo e cambio programmi su cassetta o disco per CBM 64. Dispongo di videogames come Zaxxon, Frogger 3D, Atari, Soccer, Basket "One On One" e "Tre Contro Tre" (Popeye, Pacman, Karate ed utility come Simons' BASIC, Ex-BASIC, Copy (control Load Error), Music Master, tutti originali e in Turbo Tape, a non più di L. 15.000. Massima serietà. Alessandro Liberatore - Via Ferdinando Acton, 76 - 00122 Ostia (RM) - Tel. 06/567558

Vendo e cambio programmi di tutti i tipi. Cerco programmi per le gestioni e sprogamazioni dei dischi. Rispondo a tutti. Giancarlo Ruiu - Via Giordano Bruno, 17 - 07026 Olbia (SS) - Tel. 22030

Abbiamo fondato il Commodore Club per scambi e di giochi ed utility per VIC 20 e C64. Per l'adesione mandare il nostro elenco e noi manderemo il nostro. Se hai un VIC 20 telefona allo 0522/5578280 e chiedi di Thomas. Se sei un possessore di C64 scrivi a: Alberto Castagnini - Via Canevani, 6 - 42032 Busana (RE)

Cambio programmi per C64. Posseggo oltre 150 giochi ed utility. Vendo inoltre cassetta con 50 giochi al prezzo di L. 10.000. Cerco materiale sul linguaggio macchina. Salvatore Lenza - Via C. Meditteraneo, 34 - 87029 Scalea (CS) - Tel. 0985/20177

Vendo e cambio programmi per C64 (giochi), ne possiedo circa 50. Prezzi ragionevoli. Inviare la vostra lista, ricambierò inviandovi la mia. Rispondo a tutti; l'annuncio è sempre valido. Scrivere o telefonare ore pasti a: Emanuele Silvestri - Piazza Malatesta, 2 - 47040 Verucchio (FO) - Tel. 0541/668040

Vendo VIC 20 con alimentatore, cavi collegamento e registratore più 5 cassette, ciascuna con 5 giochi, a L. 400.000. Massima serietà o Guareschi - Via Canaletto, 1 - 43010 Parma

input-output

Piccoli annunci

Compro (se vera occasione) stampante per VIC 20, larghezza foglio 210 mm minimo. Offro circa L. 200.000. Preferibilmente tratto con persone residenti nelle tre Venezie. Claudio Della Pietra - Piazza Mercato, 24/6 - Abano Terme (PD)

Dispongo di circa 400 programmi per CBM 64 e sono disposto a venderli o scambiarli. Tra il software ho anche "S.A.M." (sint. vocale), "Tot Professional", "Doodle" (ottimo programma di grafica). Telefonare solo sabato-domenica-lunedì. Renato Verdici - Via Gaspare del Fosso - 89100 Reggio Calabria - Tel. 0965/330600

Vendo per C64 cassetta con 30 programmi come: S.A.M. Reciter, Solo Flight Fox, Decathlon, Zaxxon, Burger Time, Moon Buggy, Pit Stop, Pole Position, Koala Painter, Simone's BASIC ed altri programmi originali. Tutti in Turbo Tape (compreso) a L. 50.000. Preferibilmente tratto con le province di Modena e Bologna. Marco Cremonini - Via Ponchielli, 54 - 41013 Castelfranco Emilia (MO) - Tel. 059/926078

Vendo VIC 20 seminuovo più manuali, una decina di giochi, utilità, moltissimi listati, 2 cartidge (Jupiter Lander, Scacchi), moltissime riviste, joystick stupendo, interfaccia collegabile a qualsiasi registratore a L. 230.000 trattabili. Antonio Marinelli - Via Gramsci, 123 - 25021 Bagnolo Mella (BS) - Tel. 030/620420 (telefonare la sera)

Vendo programmi per VIC 20, molti dei quali inediti, tra cui: Calcio, Grand Prix, Clow, ecc. Per ricevere listati inviare L. 1.000 a: Jerome Kubes - Via Travettore, 108 - 36061 Bassano Del Grappa (VI)

Vendo per VIC 20 programmi copiati da cartidge che richiedono espansione 16 Kbyte a L. 5.000 cadauno. Vendo inoltre schema elettrico generale del VIC 20 a L. 3.000. Telefonare dopo le 15. Enrico Cuoco - Via Milano, 125.12 - 20024 Garbagnate Milanese (MI) - Tel. 02/9959088

Vendo, cambio videogames ed utility per C64; direttamente dall'America il gioco delle Olimpiadi con 10 gare; vendo anche 23 lezioni di BASIC su cassetta a L. 50.000. Inviare l'elenco oppure telefonare a: Gianni Simone - Viale Ferruvia, 25 - 81043 Capua (CE) - Tel. 0823/96232

Per CBM 64 vendo, cambio più di 300 programmi, tutte le ultime novità, richiedo la lista. Rispondo a tutti. Stefano Castagnetti - Via Benzola, 14 - 43013 Langhirano (PR) - Tel. 0521/853123

Vendo, compro, cambio software per CBM 64. Cerco a prezzi modici Turbo Tape. Scrivere o telefonare ore pasti. Stefano Sampaoli - Via Aldo Moro, 15 - 63021 Amandola (AP) - Tel. 0736/97462

Cambio e vendo a prezzi irrisori programmi di ogni genere per Commodore 64. Alessio Alessi - Via D'Annunzio, 98 - 94015 Piazza Marmarina (EN) - Tel. 0935/80297

Vendo 10 giochi per VIC 20 inespanso a L. 9.000, in omaggio un programma listato. Assicuro massima serietà. Francesco Trimboli - Via Della Vittoria, 32 - 88028 Soverato (CZ) - Tel. 0967/21650

Vendo programmi per CBM 64. Ho quasi 140 programmi di giochi ed utility (quasi tutti in LM) al fantastico prezzo di L. 2.500 cadauno. Pagamento in contante. Per lista scrivere o telefonare a: Giovanni Tancini - Via Pian D'Ovile, 39 - 53100 Siena - Tel. 0577/288830

Vendo per CBM 64: Simons' BASIC più istruzioni a L. 40.000; data base a L. 25.000. Turbo Tape, Zaxxon, Jumpman, Billard, Master64, Sea Wolf, Sialom ed altri a L. 10.000 cadauno. Pietro Cirichetta - Via Casalicci, 28 - 73037 Poggardo (LE)

Vendo o cambio circa 200 programmi di ogni tipo per il C64 (tra questi: Magic Desk, Assembler Commodore, Simons' BASIC e molti videogames). Cerco, in particolare, S.A.M. Reciter, Easy Script e Solo Flight (solo su nastro). Inviare e/o richiedere liste a: Ivo Pecile - Via Dei Platani, 62 - 33010 Plai-no di Pagnacco (UD)

Cambio giochi per CBM 64. Ho il fantastico gioco "Ghostbusters" con una musica favolosa. Telefonare dopo le 20. Giovanni Sebastiani - Via A. Loria, 7 - 00191 Roma - Tel. 06/32712011

Cambio software per C16 e Plus. Tassati, veramente di ottima qualità. Mauro Rosso - Via Calatini, 7/E - 20052 Monza (MI)

Vendo, cambio moltissimi giochi ed utility per CBM 64 tra cui le ultime novità. Per informazioni telefonare a: Firmino Cigalotto - Via Laipaccho, 283 - 33100 Udine - Tel. 0432/284563

Costituitosi Club per utenti CBM 64. Tassa di iscrizione L. 15.000 annue. Potrete accedere ad una vasta libreria di software e molti altri vantaggi. Per maggiori informazioni telefonare ore pasti oppure scrivete. Massima serietà. Livio Guzzo - Via Palazzo, 5 - 84040 Vallo Scalo (SA) - Tel. 089/396939

Vendo per CBM 64 molti giochi a L. 1.500 cadauno. Richiedo lista telefonando oppure scrivendo. Massima serietà. Enzo De Marco - Via S. Baratta, 71 - 84100 Salerno - Tel. 089/396939

Cambio, vendo numerosi video games ed utility per C64 su cassetta. Per chi fosse interessato all'acquisto, il prezzo di 10 programmi è di L. 45.000 trattabili. Per ricevere la lista inviatemi L. 1.000 in francobolli. Annuncio sempre valido. Giulio Giannetti - Via Avv. Maccagni - 70059 Trani (BA) - Tel. 0883/47471

Vendo software di ogni tipo per CBM 64. Invio lista gratis a coloro che la richiederanno. Cerco attivamente Mario Bros per CBM 64. Roberto Spanu - Via Della Foce, 40/5 - 18100 Imperia - Tel. 0183/26121

Compro per C64 programmi per copiare giochi su nastro soltanto se a prezzi super convenienti. Scrivere o telefonare il sabato e domenica mattina a: Pietro Borgogno - Via Revigliasso, 80/6 - 10027 Moncalieri (TO) - Tel. 6406728

Vendo, causa partenza per il servizio militare, VIC 20 più espansione 3-8-16 Kbyte, cartidge "Programmer's Aid", corso BASIC (vol. 2), interfaccia per registratore, 10 cassette giochi, utilità, 3 libri, tutto a L. 280.000. Andrea Santini - Via Brioschi, 56 - 20141 Milano - Tel. 02/8465898

Vendo, cambio giochi ed utility per CBM 64. Telefonatemi solo se interessati. Brizio Zonca - Via Mora e Gibin, 69 - 28021 Borgomanero (NO) - Tel. 0322/83228

Vendo e cambio software per VIC 20. Inoltre sono presidente del Supercub Club. Spedire L. 5.000 a bimestre. Assicurato invio di software vario. Scrivere o telefonare. Francesco Capozzoli - Via Domodossola, 106 - 24091 Battipaglia (SA) - Tel. 0828/28223

Vendo ottimi videogames per C64 a sole L. 5.000 cadauno. Inviare lista. Giovanni Palumbo - Viale Kennedy - 75016 Pomarico (MT) - Tel. 751598

Cerco per C64 programmi di utility e giochi. Inoltre cerco programmi speciali: pagine rotanti con orologio, pagine con schermo diviso in 2 o in 4. Telefonare ore serali. Bruno Gandolfi - Via Calamandrei, 1 - 14049 Nizza Monferrato (AT) - Tel. 0141/727216

Per C64 compro, cambio, vendo software su nastro e cartidge. Inviare la propria lista, rispondo a tutti. Cerco floppy disk drive 1541 se affare. Cesare Storti - Via Stazione, 25 - 27100 Pavia

Vendo VCS Atari praticamente nuovo con joystick, paddle, alimentatore, 1 cassetta omaggio a L. 190.000, vendo anche diverse cassette a L. 20.000 cadauna. Telefonare ore pasti. Alfonso Di Giampietro - Via Dei Caccia, 11 - 28100 Novara - Tel. 0321/390125

Vendo, cambio software per CBM 64 su disco e cassetta, utility, gestionali, giochi, oltre 600 programmi: Magic Desk, Clone, Unguard, Decathlon, Dallas Quest, Mr. Mephisto, ecc. Marco Schito - Via Lucca, 36 - 20152 Milano - Tel. 02/4591526

Vendo programmi per C64. Tratto solo con Milano. Telefonare ore pasti. Vincenzo Arcidiacono - Piazza Gramsci, 3 - 20154 Milano - Tel. 02/317676

Commodore 64: vendo 100 programmi in linguaggio macchina a L. 100.000. Massima serietà. Claudio Rampazzo - Via Rovigo, 8/7 - 39100 Bolzano - Tel. 0471/37991

Vendo VIC 20 a L. 130.000 e regalo, a chi lo acquista, 20 videogiochi su cassetta. Valentino Magri - Via Piemonte, 15 - 20090 Seggiano (MI)

Compro, cambio vendo istruzioni e programmi per C64 e C16. Inviatemi le vostre liste e proposte. Annuncio sempre valido. Gino Ugletti - Via Strambio, 108 - 27011 Belgioioso (PV) - Tel. 0382/960804

Vendo ogni genere di software per Commodore 64 sia su disco che su nastro (in Turbo Tape). Dispongo di oltre 1.500 titoli e sono in vendita a sole L. 3.000 ciascuno con ulteriori sconti per quantitativi. Richiedete la lista a: Livian Stefano - Via Toscana, 13 - 10026 Santena (TO) - Tel. 011/9492987

Vendo giochi come: Soccer, Pacman, Alien, Star Trek, Briscola, ecc., su cassetta o listato per VIC 20 e C64. Sconti per acquisti in blocco e in più regalo Turbo Tape. Cerco Donkey Kong per C64. Scrivere a: Sebastiano Guastella - Via San Giovanni, 25 - 96010 Melilli (SR) - Tel. 955105

Vendo, cambio programmi per CBM 64 a prezzi che mi vergognerei di dire; oltre 700 games ed utility, per informazioni scrivete o telefonate a: Gianni Mazzei - Via Cella, 329 - 48020 S. Stefano (RA) - Tel. 573529

Vendo a prezzo eccezionale o cambio software per CBM 64 di tutti i generi. Scrivetemi, invierò liste programmi a tutti. Annuncio sempre valido. Italo Trave - Via F. Corridoni, 64 - 46100 Mantova - Tel. 0376/350388

Cerco disperatamente C64, solo console, in buono stato e ad un prezzo modesto. Marco Bellati - Via Progresso, 6 - 46035 Ostiglia (MN) - Tel. 31027

Vendo VIC 20 in ottimo stato più 2 cartucce, 8 cassette contenenti oltre 30 giochi (Space Invaders, Puc Man ed altri), manuale in italiano a L. 170.000 trattabili. Marco Bellati - Via Progresso, 6 - 46035 Ostiglia (MN) - Tel. 31027

Vendo VIC 20 con la sua confezione originale completa di tutto più interfaccia per registratore, paddles Commodore, varie riviste su VIC 20 e BASIC, vari programmi al prezzo stracciato di L. 140.000. Affrettatevi etto Devoli - Via Zaccari, 24 - 42048 Rubiera (RE) - Tel. 0522/620379

Compro espansione da 8 Kbyte per VIC 20 ad un prezzo ragionevole. Scrivete a: Massimo Dendena - Via Palma Il Vecchio, 16 - 24100 Bergamo

SUPERCOMMODORE

CEDOLA DI ORDINAZIONE - LIBRI
da compilare e spedire in busta chiusa a
J.s.o.f. - Viale Resetti, 5 - 20124 Milano - Tel. 6986228

LIBRI

Statistica ad una dimensione
con il Commodore 64 cod. AJAC056 L. 15.300
Commodore 64 a scuola cod. AJAN036 L. 16.200

Ordino i seguenti libri per un importo totale di L. + L. 2.000
come contributo fisso per spese di spedizione

Cod. Cod.

Cod. Cod.

☐ Contanti allegati ☐ Assegno allegato n°

☐ Ho spedito l'importo a mezzo vaglia postale

☐ Ho versato l'importo sul cc. n° 19445204 intestato a J.s.o.f. - Milano

☐ Pagherò in contantesse al postino al ricevimento dei volumi (valido solo per i soci in Italia)

Nome

Cognome

Via

Città C.A.P. Prov.

Se richiesta fattura:

Cod. F. e P. Iva

Data

Firma

Per i residenti all'estero: pagamento anticipato (vaglia o versamento su m. ccp)

SUPERCOMMODORE è bello, però... (ovvero suggerimenti, idee, critiche, richieste e tutto ciò che vi passa per la testa).

Nome
Cognome
Via C.A.P.
Città Tel.

SUPERCOMMODORE

INPUT/OUTPUT

☐ COMPRO ☐ VENDO ☐ CAMBIO
☐ VIC 20 ☐ C 16 ☐ C64 ☐ PERIF. ☐ SOFTWARE

La rubrica INPUT/OUTPUT è gratuita ed aperta a tutti i lettori. Chi desidera contribuire, vendere o cambiare hardware o software può inviare il tagliando a L.s.o.f. - Viale Resetti, 5 - 20124 MILANO

Nome Cognome
Via C.A.P.
Città Tel.

[illegible]

PERSONAL ATARI 130XE

NUMERI, NON PAROLE.

ATARI 130 XE è un grande personal computer che, ad una enorme capacità di memoria, abbina una maggiore versatilità. Atari 130XE non ha bisogno di molte parole per essere presentato, i suoi numeri parlano da soli.

128K RAM

ATARI 130XE utilizza un microprocessore 6502C. Ha una memoria di 128K RAM, 24K ROM (Sistema operativo e linguaggio programmazione Basic). Tanta potenza in più per un maggior numero di informazioni.

4 CHIPS SPECIALI

Sono Chips esclusivi Atari: GTIA - Per la visualizzazione grafica; POKEY - Generatore e controllo di suono; ANTIC - Per la gestione dello schermo e di Input /Output; FREDDY - Sistema di controllo della memoria.

256 COLORI

ATARI 130XE ha capacità grafiche nettamente superiori alla media: 11 tipi di modi grafici; 256 colori (128 visualizzati contemporaneamente); grafica 320 x 192 ad alta risoluzione; una visualizzazione di 40 colonne per 24 linee; 5 modi di testo. Tutto quanto ti serve per scatenare al meglio

la tua creatività, senza limiti di alcun tipo.

4 CANALI SONORI

Anche per le capacità sonore, Atari 130XE rivela straordinarie e inaspettate possibilità: 4 canali sonori indipendenti su una gamma di 3 ottave e mezza.

62 TASTI

Una tastiera formidabile: 62 tasti, tasto Help e quattro tasti per funzioni speciali; caratteri internazionali; 29 tasti con funzione grafica. Linguaggio di programmazione Atari Basic incorporato.

41 CONFIGURAZIONI POSSIBILI

Software e periferiche compatibili con i computers della serie XL: Stampante a colori Atari 1020; stampante "Letter Quality" Atari 1027; Stampante ad aghi programmabile Atari 1029; Disk drive Atari 1050 (È possibile l'utilizzo di 4 unità in serie); Joystick e Super Controller.

L. 380.000 + IVA



**ATARI COMPUTERS
TECNOLOGIA FORTE, PREZZO VINCENTE.**

I NOSTRI RIVENDITORI: Lazio Videosuono V.A. Baldovinetti 68/74 - Roma 06/5038525-5035880 - Lombardia + Novara Consolo & Longoni V.le dell'Industria 63 - Paderno Dugnano 02/9183372-9184083 - Piemonte - Valle d'Aosta Norvat C.so Fiume 12 - Torino 011/682171 - Tre Venezie Interservice V.S. Pietro 58/A - Padova 049/655654 - Emilia+S. Marino Telcom Via dei Mille 19 - Bologna 051/274600 - Toscana, Umbria, Marche+La Spezia Teletbit c/o Sig. Cattani Via Il Prato 8/R - Firenze 055/269652 - Puglia, Abruzzo, Basilicata, Molise Domina V. Tureti 16 - Bari 080/420204 - Sicilia, Calabria Belco V. Mariano d'Amelio 78 - Palermo 091/547566 - Sardegna Sicom Strada Sestu - Elmas Km. 2,1 Sestu - Cagliari 070/29317 - Campania Lada V. Ferrante Imperato 33 S.Giovanni a Teduccio (NA) 081/7527006
DISTRIBUZIONE NAZIONALE Canale Foto Cine Circe Film Via Pascoli 70/3 - Milano 02/2363556 - 2366410